
Electronic Thesis and Dissertation Repository

6-10-2016 12:00 AM

Improvements to Tracking Pedestrians in Video Streams Using a Pre-trained Convolutional Neural Network

Marjan Ramin
The University of Western Ontario

Supervisor
Dr. Jagath Samarabandu
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science
© Marjan Ramin 2016

Follow this and additional works at: <https://ir.lib.uwo.ca/etd>



Part of the [Computer Engineering Commons](#)

Recommended Citation

Ramin, Marjan, "Improvements to Tracking Pedestrians in Video Streams Using a Pre-trained Convolutional Neural Network" (2016). *Electronic Thesis and Dissertation Repository*. 3886.
<https://ir.lib.uwo.ca/etd/3886>

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlsadmin@uwo.ca.

Abstract

Target tracking has many applications in various fields. Millions of cameras are being used globally and people are constantly being watched everywhere. These cameras record over 48 hours of videos weekly which are impossible to be monitored manually. Many applications have been presented to improve the performance of pedestrian tracking. However, it still has remained a challenging topic. In this thesis, an automatic method is proposed for multiple pedestrian tracking. State-of-the-art detection has been combined with a tracking algorithm, followed by a novel post stage processing to increase the accuracy. Proposed automatic tracking system was compared with a state-of-the-art tracking algorithm which shows comparable accuracy when used with the original incomplete ground truth data. It is estimated to offer better accuracy with a more accurate ground truth data. The proposed algorithm offers potential improvements in both true positives as well as false negatives ratio when compared with the existing algorithm.

Keywords: Multiple Pedestrian Tracking, Deep Learning, Object Tracking

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Dr. Samarabandu for the continuous support of my Masters study and related research, for his patience, motivation, and immense knowledge.

Nobody has been more important to me in the pursuit of this project than my family. I would like to thank my adorable parents and my lovely sisters, whose love and guidance are with me in whatever I pursue and were always there for moral and emotional support although lived far away from me for past two years.

I cannot express enough thanks to my friends for their continued support and encouragement.

I also would like to thank professors at Western who provided me knowledge and guidance.

Last but not least, thank you very much for all the staff in Western University for taking good care of me.

Contents

Acknowledgements	i
Abstract	i
Acknowledgements	ii
List of Figures	v
List of Tables	viii
List of Appendices	ix
List of Abbreviations	x
1 Introduction	1
1.1 Motivation	1
1.2 Contributions	4
1.3 Document Structure	5
2 Background	6
2.1 Object Representation	6
2.2 Feature Selection for Tracking	7
2.2.1 Hand-crafted features	7
2.2.2 Features learned from data	10
2.2.2.1 Deep Learning	11

2.3	Detection	23
2.4	Object Tracking Algorithms	25
2.4.1	Target Representation and Localization	25
2.4.2	Filtering and Data Association	26
3	Methodology	30
3.1	Detection	30
3.2	Visual Tracking	31
3.3	Post Stage Processing	33
3.4	Algorithm and Software Development Steps	35
3.5	Dataset	37
3.6	Evaluation	38
4	Results	43
4.1	Occlusion Handling	48
4.2	In-house Dataset	48
5	Discussion and Future Work	55
5.1	Discussion	55
5.2	Future work	56
	Bibliography	58
A	MATLAB Implementation	65
	Curriculum Vitae	79

List of Figures

2.1	Object Representation samples [1].	8
2.2	Alex net CNN structure consists of five convolutional layers, three pooling layers, and three fully connected layers which is designed for object detection tasks [2].	13
2.3	Auto Encoder that projects the input to output.	14
2.4	Comparing a separator line in a binary space using different models. Black dots represent class 1 and blue dots represent class 2. Blue line indicates the class boundary found by various algorithms. [3].	15
2.5	Neural net before and after dropout [4].	19
2.6	Target Presentation Samples.	27
2.7	Linear Programming method for multi-object tracking [5] The image shows the x and y coordinates of bounding boxes of two moving objects. The global optimum is found using ILP although objects have overlaps in some occasions.	28
2.8	Representing ILP in a flow network where the shortest path is used to find the best possible track for each pedestrian.	29
3.1	R-CNN structure 3.1	31
3.2	This figure shows tracked pedestrians with specified label numbers as well as births and deaths of tracks of individual pedestrians.	32
3.3	Limitation of simply using euclidian distance to join the broken tracks.	34
3.4	Flowchart of the presented Post Stage Processing	40
3.5	Sample images from ETHZ Dataset.	41

3.6	Sample image from my dataset, showing multiple pedestrians.	42
3.7	Sample image from my dataset, showing pedestrians in distance.	42
4.1	(a) represents a specific frame in the tracking results of the proposed method in this thesis. (b) is the same frame which shows the result of tracking method presented by Pirsiavash et al. [6], and (c) shows bounding boxes annotated in the same frame of the original video sequence.	44
4.2	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 134 and 413 are missclassified although they are truly pedestrians.	47
4.3	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 231 and 439 are missclassified and one pedestrian could not be detected and tracked.	48
4.4	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 547 and 684 are missclassified.	49
4.5	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 135 and 51 are missclassified.	49
4.6	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 244, 312, and another bounding box are missclassified.	50
4.7	Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. It shows that the proposed method in some cases is more precise than the annotations in the original video sequence.	50

4.8	This is another sample of detecting pedestrians not annotated in the original video sequences.	51
4.9	Comparing the original video sequence, on the left, with results achieved by the proposed method, on the right. Bounding boxes 362 and 145 are missclassified although they are truly pedestrians.	51
4.10	Comparing the original video sequence, on the left, with results achieved by the proposed method, on the right. Bounding boxes 231 and 684 are missclassified although they are truly pedestrians.	52
4.11	This is a sample of equality in the pedestrians detected in both original video sequences and results of the proposed method.	52
4.12	A sequence of frames, starting from right to left, in which a pedestrian is occluded with an object. The proposed method handled this problem.	53
4.13	Partial Occlusion Handling	53
4.14	These frames show a part of the results of the second experiment presented in this study. The same approach as the first experiment on ETHZ dataset is used using same values for all the parameters like Euclidian distance threshold. . . .	54

List of Tables

4.1	This table shows the actual True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) values of the final tracking results in both Pirsiavash et al. method [6], as well as the proposed method.	44
4.2	This table shows the actual True Positive, False Positive, False Negative, and True Negative Ratio of the final tracking results in both Pirsiavash et al. method [6], as well as the presented method.	45
4.3	This table shows the actual False Negative, False Positive, as well as ID Switches values in the final tracking results of both Pirsiavash et al. method [6], as well as the proposed method.	45
4.4	Results of comparing the proposed method with the work introduced by Pirsiavash et al. [6] in terms of MOTP as well as MOTA metrics.	46

List of Appendices

Appendix A MATLAB Implementation	65
--	----

List of Abbreviations

CLEAR	Classification of Events, Activities and Relationships
CNN	Convolutional Neural Network
DBN	Deep Belief Network
HoG	Histogram of Oriented Gradients
GLM	General Linear Model
ILP	Integer Linear Programming
LDCF	Locally Decorralated Channel Features
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
RNN	Recurrent Neural Network
R-CNN	Regions with Convolutional Neural Network features
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine

Chapter 1

Introduction

1.1 Motivation

Nowadays, a large number of cameras are being used in various places, and the use of such recording devices is increasing dramatically. Hence, monitoring all these cameras by human operators is not possible anymore. When looking at the applications of tracking related to humans (pedestrians), accurate tracking is an essential need. Visual tracking is a challenging problem which has applications in a number of different fields of research such as biomedical imaging, video surveillance and robot navigation. Safety is crucial in our lives and much attention has been paid to this issue. We need to remain safe everywhere, while driving on the road or even walking down the street. In 2010, about 270,000 pedestrians were killed on the roads globally, which is about 22% of the 1.24 million deaths in traffic accidents, which shows the importance of investigation of different approaches to reduce traffic fatalities. One way to decrease the number of car accidents with pedestrians is to equip vehicles with cameras that detect and track pedestrians on the road. Great improvements have been achieved in face detection and tracking using professional cameras. However, these kinds of cameras are very expensive and cannot be used widely. Different computer vision algorithms are used in various cameras to analyse the video streams for specific tasks. However, usually, they cannot be gen-

eralised for analyzing various types of video streams and different tasks. For example, those professional cameras which can perform the face detection tasks are not suitable to monitor people walking in the street since these kinds of cameras are expensive and are not equipped with enough memory space to record hours of videos. Besides, face detection cameras fail in some occasions including distance-based failures, crowd scene failures, etc. [7]

Although many studies have been done leading to a considerable progress during the past decade, visual tracking still remains a challenging topic due to the numerous factors that affect its accuracy. Variations in viewpoints (camera positions), occlusion, variations in light (illumination), as well as camera distortion, are among them. Automatic pedestrian tracking is an important need for many applications including security systems in crowded places like airports, autonomous vehicles and even in intelligent sports analysis. Pedestrian tracking remains an active area of research in computer vision due to limitations of current methods. There are many factors that make a tracker perform well including being insensitive to camera motion, low contrast level, occlusion, fluctuations in illumination and number of pedestrians visible in the video frames. There are many conditions that make the tracking tasks very challenging and difficult. For instance, a tracking algorithm may perform up to an acceptable level in appearance variations. However, there may be considerable inaccuracy due to variable illumination. Tracking algorithms can be categorized into two subcategories of single object tracking, and multiple object tracking. In the case of having multiple objects or pedestrians in the scene, the problem would be more complicated compared to single object tracking. This is due to the fact that challenges like occlusion and object interactions do not exist in the videos where only one object is visible and needs to be tracked. A number of issues, which affect the accuracy of object tracking are listed below [8, 9]:

Camera motion Two types of cameras (fixed and moving) can be used for recording videos. In the videos captured by moving cameras, changing in object positions are more complicated leading to more difficult tracking problems.

Complex motions The complex motions that objects/pedestrians can have in video frames can affect the tracking results, and make it a more complicated task. For instance, tracking hockey players who have very unpredictable motions during the game would be much more difficult compared to the tracking of a vehicle going on a straight street.

Variations in illumination level Change in the appearance of background and objects/pedestrians due to the variations of illumination can affect the detection accuracy, which in turn will affect tracking.

High and low density The density of people appearing in the video significantly affects results in different pedestrian tracking algorithms. For an example, in videos with a comparatively higher density of pedestrians, occlusion of other pedestrians poses a significant challenge. However, in videos with low densities of people, such occlusions are rarer. Also, in most of the cases, the image of the full body of each pedestrian can be captured. There are other challenges associated with densely crowded scenes including pedestrians with very small sizes, and difficulty in detection of each individual due to spatial compactness, as well as complexities in human interactions.

Single and multiple object tracking In multi tracking problems, multiple tracks must be followed carefully to prevent missing objects/pedestrians in terms of occlusion while they are crossing each other.

There are several factors which help improve the performance in object tracking:

1. Smooth motion without sharp changes.
2. Having a fixed camera instead of a moving platform.
3. Gradual changes in background.
4. No sudden variation in object appearance.

5. Small number of objects of interest.
6. Limited amount of occluded objects.

1.2 Contributions

In this work, one of the most difficult and challenging types of tracking scenarios is considered. This scenario involves tracking multiple pedestrians in a crowded scene, recorded by a moving camera, in a real world environment with variable illumination. An automated tracking algorithm is proposed in this thesis using state-of-the-art detection and tracking algorithms following a post stage processing to increase the accuracy. My results show that using a pre-trained convolutional neural network in the detection phase leads the algorithm to be accurate enough for pedestrian tracking in a crowded scene. This allowed the tracking results be independent of factors like camera movements or pedestrians distance from the camera. Taking advantage of deep learning for detecting pedestrians with different appearance and situations makes the proposed method more robust, since, pre-trained deep neural network presented by Tom et al. [10] finds pedestrians with a high accuracy. Further, in certain scenarios, use of Deep Convolutional Neural Networks are capable of delivering more precise results than human eyes in cases that it is difficult to detect pedestrians visually. In terms of tracking performance, the presented method is comparable with the tracking algorithm presented by Pirsiavash et al. [6].

- In this thesis, an investigation was carried out on using Deep Learning as detection and classification phase in a pedestrian tracking problem. A raw video stream was fed to a fully automated algorithm for multiple pedestrian tracking without any preprocessing step.
- A novel post stage processing algorithm is proposed to enhance the tracking algorithm developed by Pirsiavash et al. [6]. In post stage processing, for each video frame at time t , next three sequential frames are processed in order to find the best matches, among the

broken tracks produced from the tracking phase. A combination of Euclidian distance and a second-order feature is used to find the best matches.

The reason that proposed algorithm seems to only match the performance of the existing tracking algorithms is that the annotated video sequences used as ground truth used in both the proposed method as well as the tracking algorithm presented by Pirsiavash et al. [6], contains a significant amount of missing labels. If these missing labels can be added to the annotated video sequences and used as ground truth, the proposed algorithm is estimated to produce better results. These results will be submitted to International Conference on Information and Automation for Sustainability - Dec 2016 (IEEE).

1.3 Document Structure

Remainder of the thesis is organized as follows: A literature review is presented in chapter 2 which describes different feature extraction methods as well as detection and tracking algorithms. In chapter 3, the methodology of the proposed method is described in detail. It contains descriptions of the detection and tracking phase as well as the post stage processing which is used to improve the performance. Chapter 4, contains detailed information about the results achieved by the proposed method. Results in different situations are illustrated in this section. In chapter 5, the results are discussed in detail and suggestions for future work are made that may improve the results.

Chapter 2

Background

In this chapter, a brief overview of concepts that are used in the proposed research is presented. These concepts include object representation, feature selection, detection, and object tracking algorithms.

2.1 Object Representation

For a tracking algorithm, objects could be anything that can be used for further processing, like a ball rolling on the ground, soccer players running in the field and a pedestrian crossing the road. Objects shape and appearance can help to categorize that object to the corresponding type of objects. Objects can be represented in following ways [1]:

1. Points. In case of having small objects in an image, point representation is a good choice, since objects can simply be presented by either their centroid points or a set of points (figure 2.1, a and b).
2. Primitive geometric shapes. Objects shapes are represented by geometric primitives such as rectangle, ellipse, etc. This representation is suitable for objects which can easily be approximated by a geometric shape (figure 2.1, c and d).

3. Object silhouette/contour. The boundary of an object corresponds to its contour, and the inside region of a contour is the object silhouette. This method of representation is suitable for objects that have an irregular shape that cannot be adequately represented using a geometric primitive (figure 2.1, g, h and i).
4. Articulated shape model. In this model of representation, different parts of objects are defined using simple shapes, like ellipse, which are joined together to create the object shape (figure 2.1, e).
5. Skeletal model. It shows the anatomical shape of an object, and it mostly used when tracking human movements (figure 2.1, f).

2.2 Feature Selection for Tracking

2.2.1 Hand-crafted features

Hand-crafted features have successfully been used in pedestrian detection. Hand-crafted features are extracted from images based on the algorithms which are pre-defined manually for certain tasks [11]. In the work presented by Wang et al. [12], the problem of occlusion and partial occlusion is investigated using Local Binary Pattern (LBP) and Histogram of Oriented Gradient (HOG) features. In another approach, presented by Felzenszwalb [13], a mixture of local templates are considered to solve deformation in pose and view. In a research presented by Benenson et al. [14], both channel features, as well as depth information, are used feature extraction. Integral Channel Features [15] and Aggregated Channel Features [16] are also other kinds of features used in object detection. Hand-crafted features can be followed by a classifier

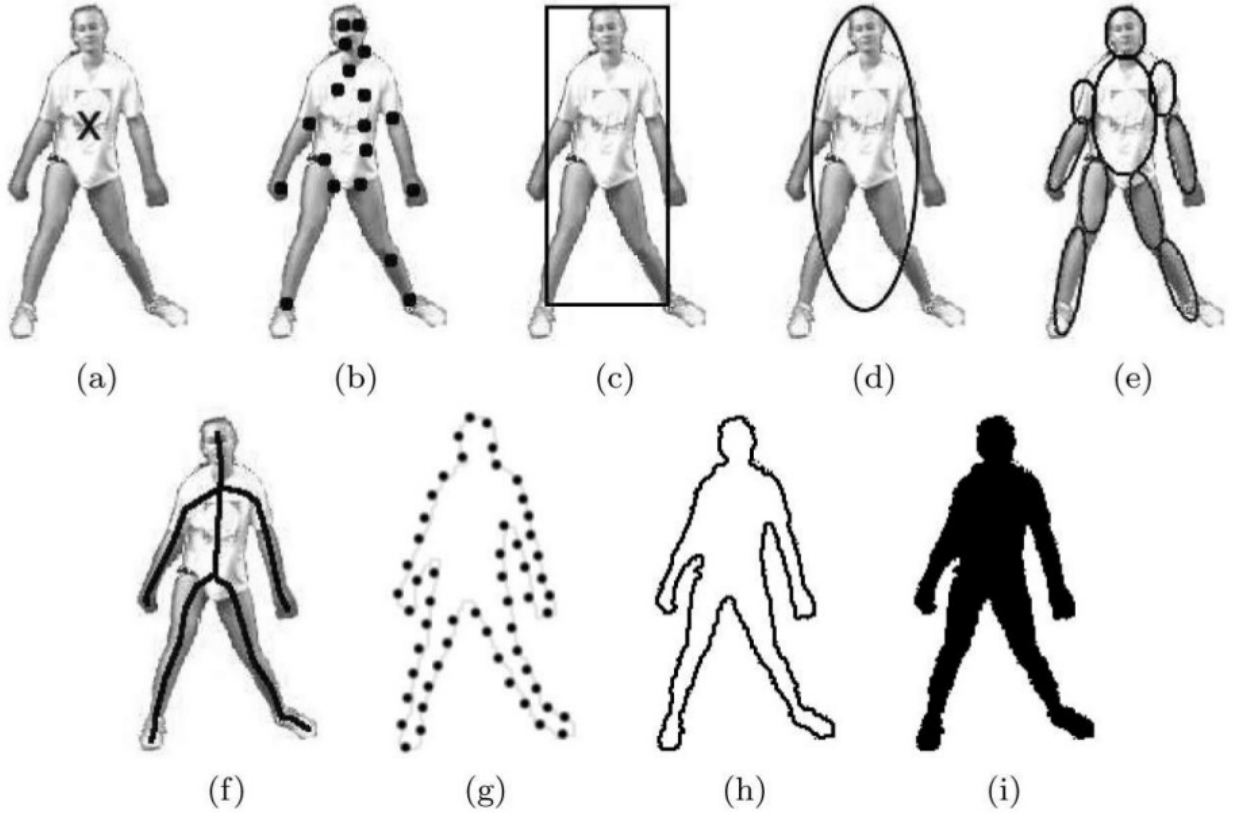


Figure 2.1: Object Representation samples [1].

such as boosted classifiers [17], random forests [18], or Support Vector Machine [13, 19].

Using statistical texture descriptors helps extract data efficiently from large datasets while keeping the whole data. Statistical approaches represent the texture using the distributions and relationships between the gray levels of an image [20].

- First-Order Statistics

These features represent the first-order statistics from the gray-level intensity histogram. In the formulation (2.1), I is the symbol of gray-levels of the image region. $p(I)$ is the probability of the graylevel I .

$$p(I) = \frac{\text{number of pixels with gray level } I}{\text{total number of pixels in the region}} \quad (2.1)$$

The Mean, Variance, Skewness, and Kurtosis of gray-level in an image is defined in (2.2), (2.3), (2.4), and (2.5) respectively.

$$\mu = \sum_{I=0}^{N-1} Ip(I) \quad (2.2)$$

$$\sigma^2 = \sum_{I=0}^{N-1} (I - \mu)^2 p(I) \quad (2.3)$$

$$s = \frac{1}{\sigma^3} \sum_{I=0}^{N-1} (I - \mu)^3 p(I) \quad (2.4)$$

$$k = \frac{1}{\sigma^4} \sum_{I=0}^{N-1} (I - \mu)^4 p(I) \quad (2.5)$$

N is the number of possible gray levels.

Variance measures the deviation of gray-levels from the Mean. Skewness represents the degree of histogram asymmetry around the Mean, and Kurtosis measures the histogram sharpness [21].

- Second-Order Statistics

The features extracted by first-order statistics represent information about the distribution of gray-levels in an image and do not provide any information about the position of gray-level values [21]. These relations can be represented by a Co-Occurrence matrix that consist of values which provides how many times two pixels with gray-levels I_1 and I_2 appear in the window with distance d and direction θ .

Different features can be extracted from the Co-Occurrence matrix which define the texture of image subregions. Contrast, correlation, entropy, and homogeneity are the most important features that can be extracted from the Co-Occurrence matrix using equations (2.6), (2.7), (2.8), and (2.9) [21].

$$Contrast = \sum_{I_1, I_2} |I_1 - I_2|^2 \log p(I_1, I_2) \quad (2.6)$$

$$Correlation = \sum_{I_1, I_2} \frac{(I_1 - \mu_1)(I_2 - \mu_2)p(I_1, I_2)}{\sigma_1 \sigma_2} \quad (2.7)$$

$$Homogeneity = \sum_{I_1, I_2} \frac{p(I_1, I_2)}{1 + |I_1 - I_2|^2} \quad (2.8)$$

$$Entropy = \sum_{I_1, I_2} p(I_1, I_2) \log p(I_1, I_2) \quad (2.9)$$

Contrast is a measure of local variations which has high values for images with high contrast. Correlation measures the correlation between pixels in two directions. Homogeneity has a large value for images with low contrast. Entropy represents the randomness and has high values for sharp images.

2.2.2 Features learned from data

Features also can be extracted using different kinds of machine learning algorithms. First, a short description is needed about machine learning algorithms. Machine learning algorithms are usually categorized into two types: supervised learning and unsupervised learning.

1. Supervised learning:

In the supervised learning methods, a mapping from input to output is learned using labelled datasets. In classification problems, the goal is to learn a mapping from input to output, where the output could be simply two classes which is called a binary classification, or more than two classes which is known as multi-class classification. Supervised

learning algorithms have two main steps:

(a) Training:

In the training step, a classification model is constructed by examining the training dataset provided.

(b) Testing:

In this phase, totally new and unseen instances are classified using the model built in the training step.

For instance in a classification task, a software is used to examine a set of images to see whether they are human or not. In this problem, the input is a set of images containing objects as well as images of people, and two classes (person and not-person) are the outputs. A set of features or attributes stored in a matrix are used for this purpose. Moreover, there is also a training vector containing two values of 0 and 1 corresponding to the classes of person, or not-person. It needs to be generalized beyond the training set in order to classify the attributes into person and not-person classes.

2. Unsupervised learning:

Unsupervised learning algorithms are more based on human learning system which is based on finding a hidden pattern in the input dataset. In this model, input data is unlabeled. Clustering algorithms are the most common methods in this model.

2.2.2.1 Deep Learning

Many applications of deep networks are introduced during the past decade. They can be used for feature extraction, classification, regression, dimension reduction, etc. For classification and regression a deep neural networks is similar to a common neural network. However, the

difference is in the number of hidden layers and usage of different regularization techniques. On the other hand, different structures for different applications are introduced. For example, Convolutional Neural Networks (CNN) [22], Autoencoders [23], Deep Belief Networks (DBN) [24], and Recurrent Neural Networks (RNN) [25] are different types of deep neural networks.

2.2.2.1.1 Deep Learning Applications In this section, different applications of deep learning are discussed.

- **Feature Extraction**

Deep learning algorithms in computer science can be used for feature extraction. The main point of using such complex model is that it has the capacity to learn how to extract informative features. Two types of deep learning algorithms used for feature extraction are described below.

1. **Convolutional neural network**

A Convolutional Neural Network (CNN) is a type of deep learning model containing different types of layers which performs feature extraction as well as classification tasks. Designing a good method for feature extraction is challenging and depends on the problems and cannot be generalized easily. It means, in case of variations in the problem or images being processed, different approaches should be used for feature extraction.

One of the most important aspect of deep learning is that the whole image can be mapped into a Convolutional Neural Network. This model can get an image as an input and predict the corresponding classes from the raw pixels.

A famous deep CNN structure is Alex-net [2] that shows a very good performance on image classification. Alex-Net consists of five convolutional layers and three

fully connected layers. It computes robust features using various filters over all the layers to classify a very large dataset containing 1.2 million images into 1000 different classes where they were able to decrease the classification error rate from 26.1% to 16.4%. Figure 2.2 shows the structure of this deep neural network. The first layer is a 224×224 image in which all the raw pixels are directly mapped to the network. In this model the first 5 layers are convolutional and pooling layers, and they are responsible for feature extraction.

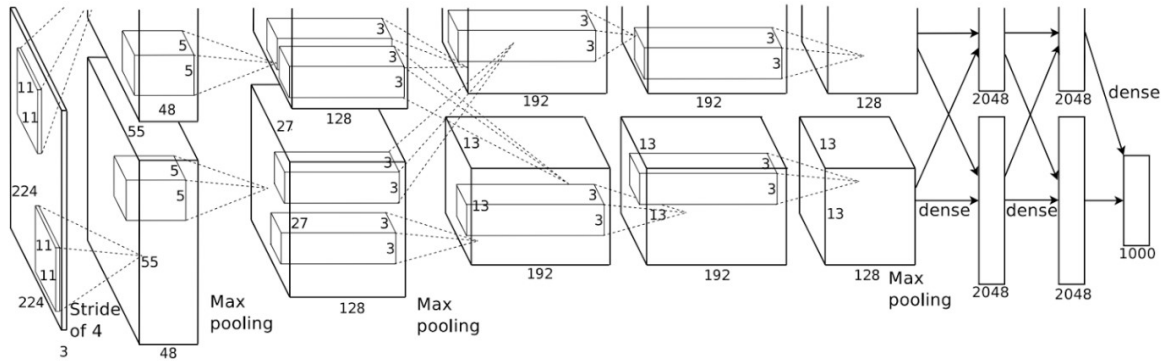


Figure 2.2: Alex net CNN structure consists of five convolutional layers, three pooling layers, and three fully connected layers which is designed for object detection tasks [2].

2. Deep Autoencoders

Autoencoder is another application of deep neural networks, which tries to predict inputs. In this model, after the network is trained, features can be extracted from the middle layers which have lower number of nodes. Autoencoders are unsupervised learning algorithms used for dimensionality reduction. This model is used for feature extraction which helps to solve overfitting in the classification problems. In unsupervised learning labels are not used during training, thus, for training an autoencoder, labelled set of data cannot be used. The idea here is to let a neural network predict its input, and be trained. Then, use the weight of this trained network for initialization. In an ordinary neural network, a network is initialized with

some random weights while in training a network using auto encoders, the weights of trained network can be applied for initialization. It also helps to decrease the overfitting problem in the classification tasks.

For example, to train a deep neural network with 6 layers, an unsupervised approach is used to train the network to predict the input. Then the obtained weights are applied for initialization, and start to train the network again. However, this time labelled set of data is used in a supervised manner to classify data using features extracted by Autoencoders. Another type of autoencoders is Denoising Autoencoders in which noise is added to the input, and that helps to reduce overfitting. Auto encoders are considered as one of the deepest neural networks with published results, ranging between 7 and 11 layers [26].

In the example shown in figure 2.3, the weights can be extracted from the centre

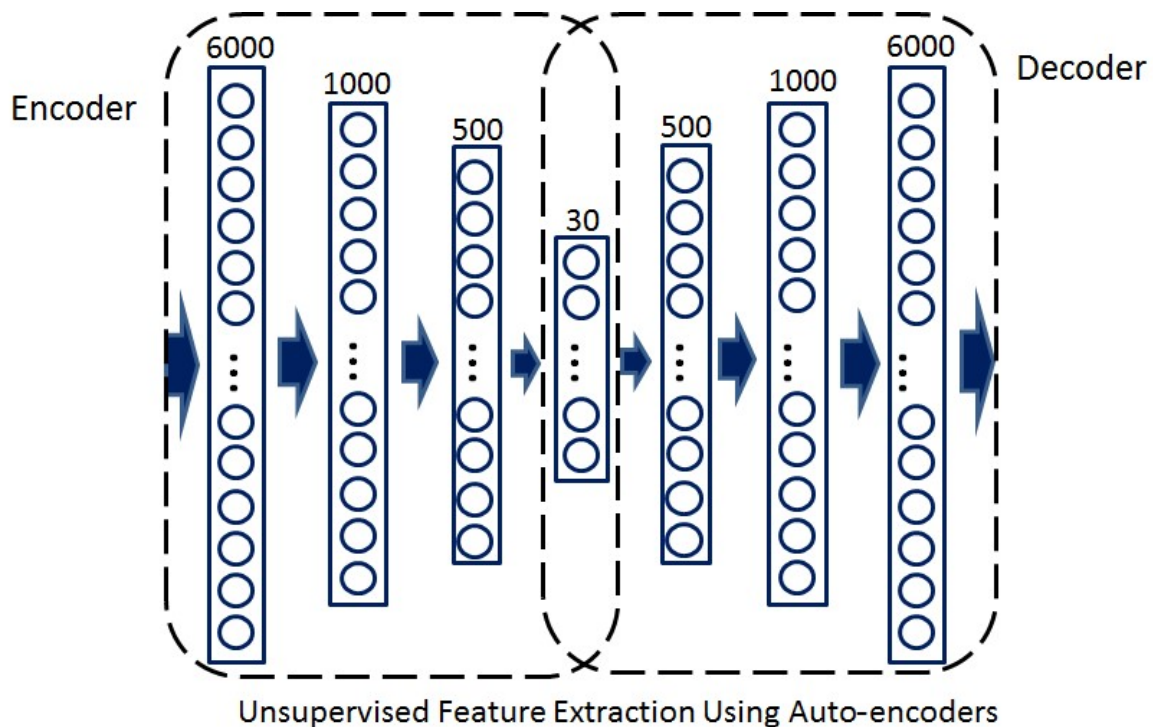


Figure 2.3: Auto Encoder that projects the input to output.

of the hidden layer after reducing dimensions from 6000 to 30.

- **Classification and Regression**

Deep neural networks can be employed for classification and regression problems. In this model, besides of using several hidden layers, regularization and drop out are applied as well. In this model, deep learning is used for classification and regression as a way to find a non-linear boundary in the data space. In an online article by Rickert [3], an example is presented on using a data set that is distributed in the shape of a spiral (see figure 2.4), and compares deep learning with Gradient Boosting Model (GBM), Decision Random Forest (DRF), and Generalized Linear Model (GLM). As can be seen in figure 2.4 , the data need a nonlinear boundary. GLM failed to classify data, since it tries to find a linear line to separate data. The other methods could classify data into two classes. However, deep learning is able to find more accurate boundary compared to the other models.

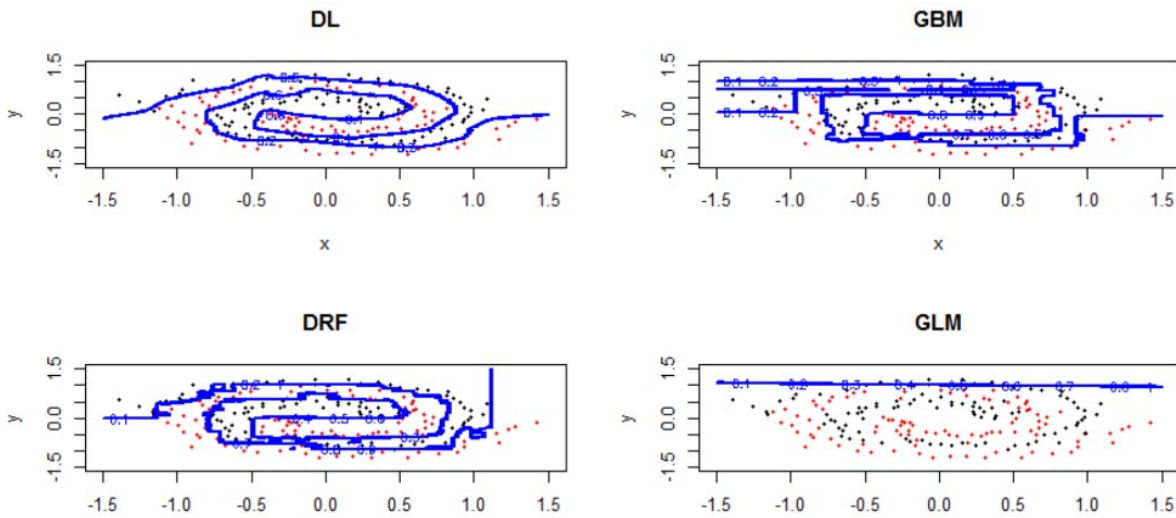


Figure 2.4: Comparing a separator line in a binary space using different models. Black dots represent class 1 and blue dots represent class 2. Blue line indicates the class boundary found by various algorithms. [3].

2.2.2.1.2 Deep Learning Parameters In this section different parameters of a deep neural networks are described.

1. Number of hidden layers:

Number of hidden layers is an important parameter in all kinds of neural networks as well as deep neural networks. Selecting the right number for this parameter is really important and affects the results directly.

2. Number of nodes in hidden layers:

Considering a large number for this parameter helps achieving generalization. Thanks to regularization, selecting a big number may not lead to overfitting, however, it needs more computation time for feeding and backpropagation.

3. Regularization:

Although Deep Neural Networks are powerful models, they have a good potential for overfitting which occurs when the network is very complicated with too many parameters, leading the network to memorize data instead of learning them. Regularization is an approach used to reduce overfitting. Two types of regularization are used for deep neural networks:

L2:

$$L2 = \lambda \sum_i \theta_i^2 \quad (2.10)$$

where θ is the weight of a hidden layer and λ is the coefficient of regularization, which indicates how much regularization should be added. λ is a parameter that needs to be tuned and finding the best value is a tradeoff between bias and variance. The square of weight value is penalized in a layer. This tries to shrink the weight to be zero with a square penalty. This can be applied to the weight only, not on the biases. This guarantees

that despite a strong regulation, bias weights are optimized. $L2$ is also known as Gaussian prior, and the reason is that it is assumed that the weights of neural networks come from a Gaussian prior. This means that the weights of a network follows a Gaussian distribution.

$L1$:

$$L1 = \lambda \sum_i |\theta_i| \quad (2.11)$$

where θ is the weight of a hidden layer and λ is the coefficient of regularization which indicates how much regularization should be added. λ is a parameter that needs to be tuned, and finding the best value is a tradeoff between bias and variance.

In $L1$, the sum over square value is replaced by the absolute value. Therefore, the absolute value of the weight would be penalized. The difference here is $L1$ can push some weight to be exactly zero. Therefore, by using this regularization, network learns which neurons should be connected to each other. When the connection between neurons are removed, the network would be less complex and therefore less flexible. This property results in preventing overfitting in the training data. In addition, $L1$ is used for input layer, because the weights can be set to zero in $L1$, it can do filtering for inputs. Sometimes some variables in the input are not useful, and $L1$ regularization can act as a feature selection to remove those useless variables.

Two different parameters as coefficients are needed for $L1$ and $L2$ regularization. One reason of applying different approaches for input and output layers is that these layers might be sparse. In the input layer, some variables are zero most of the time, and in the output layer some neurons may correspond to a class or an event that happens rarely.

4. Dropout:

Combination of different models in some cases can lead to a better model. For exam-

ple, in random forests, combining different trees that are weak classifiers helps to make a strong model. Similar method can be used for neural networks and train several of them, and make a stronger model. However, for combination of neural networks some problems may occur. Training neural networks is computationally expensive, thus, these kinds of combination models cost a lot. In addition, different structures have to be defined for each network, and tune them separately. It is difficult to tune a single neural network, therefore tuning several of them is a demanding task. Furthermore, similar to random forests, different training subsets are required for each model. However, in deep neural networks, each sub-model is a neural network which needs many training data. Thus, the training data is reduced in a model that needs huge amount of training data.

Drop out is a method which allows us to benefit from combination of different neural networks without facing these problems. Considering a deep neural network with 10 layers, each layer is a neural network and needs to be combined with other layers which are independent neural networks as well. Similar to random forests, they can be made by naive predictors by disabling some nodes in each layer.

Because there is a complicated relationship between input and output, a deep neural network can be overfitted easily. Dropout is introduced to solve overfitting problem. While $L1$ and $L2$ focus on input and output layers, dropout is designed for all layers including hidden, input, and output layers. The idea is simple, and some units are required to be selected from a certain layer and then be removed from the network. By removing nodes from the layer, that node is disconnected from previous and next layers (see figure 2.5). In order to do this, a probability need to be assigned to each node that determines whether it should be removed from the network or not. For example 0.5 for a certain layer means that each unit has 50% chance to be removed from the network. Therefore, in each layer a probability number is required to be found that determines the chance of removing each unit (neuron) from the network. A network with 5 layers needs 5 numbers to indicate this probability. A value of 0.5 works in many applications, and by applying

it, roughly half of neurons from the corresponding layers are expected to be disabled. This number can be found by cross validation and other methods used for tuning [4]. For training a network in which some neurons are disabled, stochastic gradient descent is used similar to a standard neural network. It can be considered that after dropout, a thinned network can be shaped, and forward and backpropagation can be applied on this new thinned network. A special activation function is designed for the dropout that will be discussed in the next section.

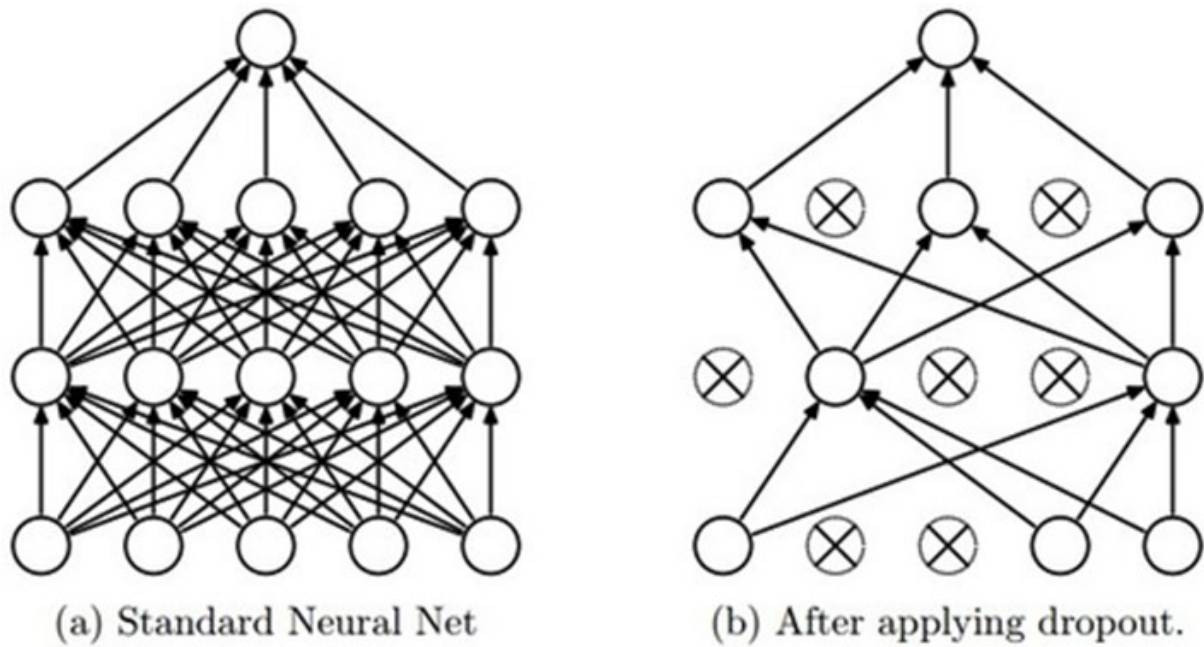


Figure 2.5: Neural net before and after dropout [4].

5. Activation function:

The typical form of output in one layer is

$$a = s(w'x + b) \quad (2.12)$$

where w is a vector of neuron weights, x is a vector of variables, and b is bias. s is a function that determines whether neurons are active or not. Using sigmoid function

as activation function is very common in the classification networks. However, different non-linear functions also can be used for deep neural networks including hard tanh, rectifier, and Maxout. Each of them has different properties, and here two of them are described briefly.

- *Maxout*

This function is designed for dropout. The idea is to take the maximum of inputs for each neuron in hidden layers:

$$\text{Max}(w'x + b) \quad (2.13)$$

This idea does not work for all types of neural networks, and it works only for feedforward neural networks (e.g. multi-layer perceptron and CNN).

- *Tanh*

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.14)$$

Tanh maps output between -1 and 1. Due to the fact that *Tanh* output is symmetric at 0 (is between 1 and -1), the training is better for a network with many layers. It is worth mentioning that if these types of functions are combined, any non-linear function can be created.

6. Learning rate:

This is one of the important parameters in deep learning. Finding a good value for this parameter is crucial when Stochastic Gradient Descent (SGD) is being used. In SGD each data sample is used separately during the training, while in gradient descent all training data is used in each iteration. For example, in a problem with 100 training data, each single sample can be fed separately to the network in each iteration. When all 100

data are given to the network once, it is called an epoch. Because a single data is used in SGD, the update is much more variant. Therefore, a small learning rate like 0.01 is required in SGD which works on standard multi-layer neural networks, to reduce fluctuation. However, definitely finding a good value for it can improve the result. Several techniques have been developed for a flexible learning rate to change the rate during the training. Schedule rate can be used as a technique in which learning rate changes during the training. A very simple schedule is to set learning rate 0.05 at the beginning, then change it to 0.01 when the error become less than a fixed threshold, or decreasing the rate after n iteration. Another common schedule can be achieved using equation (2.15).

$$\text{learning rate} = \frac{i}{b + t} \quad (2.15)$$

where t is the iteration number, i is initial learning, and b is the number of iterations that the algorithm needs to converge.

7. Number of training iterations:

This one is also considered as an important parameter. It should be large enough to let the algorithm to coverage. Although, maximum number of iterations is not a tuning parameter in a simple neural network, it is important for a deep one. In a deep neural network, this parameter can be used as a technique to prevent overfitting. This technique is called early stopping, and the idea is to stop training before that network converge on the training set. Early stopping, helps to prevent overfitting problem in deep neural networks. In this technique, validation is used to calculate the error during the training. As long as the validation error is decreasing sharply, it can be continued. Therefore, it shows how much iteration is enough by looking at validation error instead of training error. Therefore, training need to be stopped before convergence on training set.

2.2.2.1.3 Tuning Parameters Deep learning has many parameters for tuning, and several methods are used for this purpose. A validation set can be helpful to determine which parameters can lead to a better performance. Although, for this complex model, there is not a direct relation between validation error and test error, using a validation set is the best option. Therefore, considering a validation set is used, the following methods are available for tuning hyper-parameters.

- **Coordinate Descent and Multi-Resolution Search**

This is a manual search for tuning different models, and this can be done in two ways: First is to fix parameters, and adjust them one by one. The second approach is to try to change all parameters at the same time. The idea of multi-resolution is to start with a big range of number, then limit the range. For example, 100, 50, 10, 1 can be tried out and select the best ones, then limit the range.

- **Automated and Semi-automated Grid Search**

The idea of grid search is to try all combinations of parameters. Using parallelization and clustering, best possible parameters can be found. However, it is usually good for less than 4 parameters. Humans have shown a good performance in tuning the parameters, therefore human can help grid search to be semi-automated. The idea of semi-automated grid search is to use multi-resolution search to guide grid search. Thus, a reasonable range can be found by manual search, then the grid search can be performed on founded ranges.

- **Random Sampling of Hyper-Parameters**

The idea of random search is to select a range for each parameter and select a number randomly for each parameter. In many partial applications, if there are some good values for Hyper-Parameters, random search is likely to exploit it [27]. Thus, random search can find them without paying an exponential price (grid search). The big advantage of random search is that it is suitable for parallelization. Therefore, different clusters and

servers can be used to find the best parameters.

Random search can be Semi-automated by guidance of human. In this way the random range can be changed after running a few experiments according to the best result.

- Other methods

Some optimization algorithms are introduced that work better than random search. For example, Bergstra et al. [28] claimed that their algorithm is better than random search, however it works only for a special type of deep neural networks, deep belief networks.

2.3 Detection

Object detection is the main part of object tracking phase which plays an important role in the final results and directly affects the performance of tracking.

- Point Detectors:

In point detection methods, interest points with different properties, like corners and edges, are detected in an image instead of a whole image. Then this detected interest points can be used for further image processing tasks. Interest points in a blob have the same values in terms of color and brightness, and are different from their neighbor pixels. This method is more useful in image matching applications [29]. Moravec's Detector and Harris Detector are some examples of this type of detectors.

- Segmentation:

Segmentation methods are used in the image processing applications in order to locate objects of interest in an image, and represent their boundaries. In this method, objects are segmented based on their similarity in the characteristics like color, texture, and pixels intensity [30]. Mean shift clustering and active contours detect objects of interest using

segmentation.

- **Background Modelling:**

It is a technique in image processing field which is based on extracting objects of interests from the background. This is used in case that the locations of objects are required for further processing. In this method, a reference or background model is required in order to extract the subtraction of current frame with it. This method is used for detecting moving objects in videos captured with static cameras and it is not applicable to the videos recorded by moving cameras, hence, it cannot be used in the real environment problems [31].

- **Supervised Learning:**

Different machine learning algorithms like SVM, Neural Networks, and adaptive Boosting can be used in detection problems.

- **Deep Models:** Many successful results have been achieved in pedestrian detection using Deep Learning methods. This success owes deep learning's ability in extracting proper features from raw images without any pre-processing. There have been many publications using deep learning for object/pedestrian detection in recent years. For instance, the JointDeep model [32] designed a Convolutional Neural Network with a hidden layer in which jointly learns feature extraction, part deformation handling, occlusion handling, and classification. In ConvNet [33], an unsupervised pre-trained CNN is used for pedestrian detection. In another work, Quyang [34] introduced a method for occlusion handling in pedestrian detection considering visibility of body parts at multiple layers. In the work presented by Tian et al. [35], TA-CNN jointly learns features from multiple datasets and tasks.

2.4 Object Tracking Algorithms

Object tracking is a crucial task in computer vision applications like car navigation, surveillance, etc, containing two major steps of detecting the moving objects in each frame in a separated detection phase, and tracking them in another step. Object tracking, usually, starts by detecting the objects of interests in each frame in a separate phase, and suggest a big set of detected targets, then finding the correspondence objects across frames. The task of joining corresponding detected targets to create tracks is called data association.

2.4.1 Target Representation and Localization

It is a bottom-up algorithm to track complicated objects with complex interactions with other objects like occlusion.

- Point Tracking

Tracking can be accomplished by considering detected objects which are defined by points in sequence of frames. In this model the connection of the points are defined based on the previous position and motion of the object (Figure 2.6, a).

1. Kalman Filter: (This is an equation to estimate the process status in past, present, and future.) This is an Optimal Recursive Bayesian filter which uses a series of observed measurements, containing statistical noise, and presents estimation of unknown status. It is a two-stage filter in which in each iteration performs a prediction for the current location of the moving object based on its previous location.
2. Particle Filter: (This a recursive Bayesian estimator which calculates the posterior distribution.)

3. **Multiple Hypothesis Tracking:** In this method, all possible tracks are calculated at first, then tracks with low probabilities are excluded using different filtering methods. This approach is really time consuming, and requires a large amount of memory.

2.4.2 Filtering and Data Association

It is a top-down algorithm for object tracking using different tools, usually with low computational complexity.

- **Kernel Tracking**

This model of tracking corresponds to the shape and appearance of the objects in which the kernel can be a rectangle or ellipse. The object motions in a sequence of frames are computed to track the desired object. Motion usually refers to transformation forms namely translation, rotation, and affine (Figure 2.6, b).

1. Simple Template Matching
2. Mean-Shift Method
3. Support Vector Machine

- **Silhouette Tracking**

This model can be considered as a segmentation method which tracks the objects by estimating the region of the objects in the frames. It is used when shapes of objects are complex and it is not possible to describe them by simple geometric shapes. The most important advantage of using silhouette tracking models is the ability of this model to handle various shapes of objects (Figure 2.6, c and d).

1. Contour Matching

These approaches estimates an initial contour of the object in its new region in the current frame. It needs that object in the current frame have a partial overlap with the object in the previous frame.

2. Shape Matching

These approaches search the current frame to find the object silhouette. The search is based on the similarity of the object in the current frame with the generated model using the hypothesized object silhouette on the previous frame.

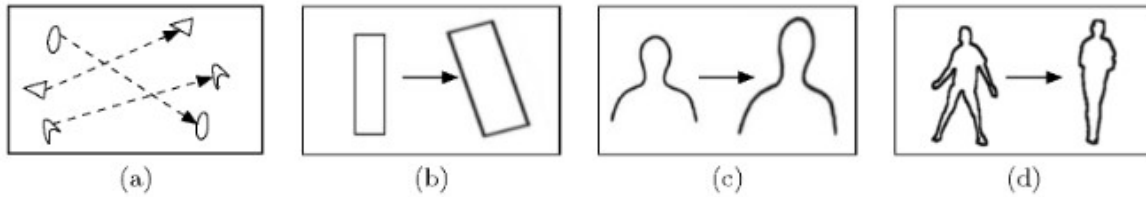


Figure 2.6: Target Presentation Samples.

One of the methods which have good performance in multi-object tracking, is tracking objects frame-by-frame, or consider a short period of time in the sequences [36, 37]. However, to achieve better results in case of long-term occlusions, and decrease the ratio of false positives, and miss detections, a longer period of time can be considered in the frame sequences. It means that more frames can be processed to solve the multi-object tracking, instead of simply looking at one frame at a time. There are also many other works in which more global information is used [38, 39, 40]. However, their searching spaces increase in size by increasing the number of frames. Therefore, there should be a pruning strategy to limit the search space. Besides, they consider all the detected objects are true positives, which it is not always the case.

In the study presented by Pirsiavash et al. [6], it is assumed that input is a video sequence of

frames with bounding boxes (generated by pedestrian detection algorithm). However, in their implementation which the proposed method is compared with, they have used a part-based HOG pedestrian detector [41] for the detection phase. Then, they applied Hungarian Bipartite Graph Matching [42], to create the tracklets by finding the shortest path based on different measures such as predicted position similarity and last position similarity measures.

The main problem of this method is that although it is locally optimal, it may not be globally optimal due to the fact that greedy algorithms are not guaranteed to achieve the global optima. In order to find a solution which is globally optimum, Linear Programming (LP) [43] is used to produce a finite set of objects from the potential objects in the frames (Figure 2.7).

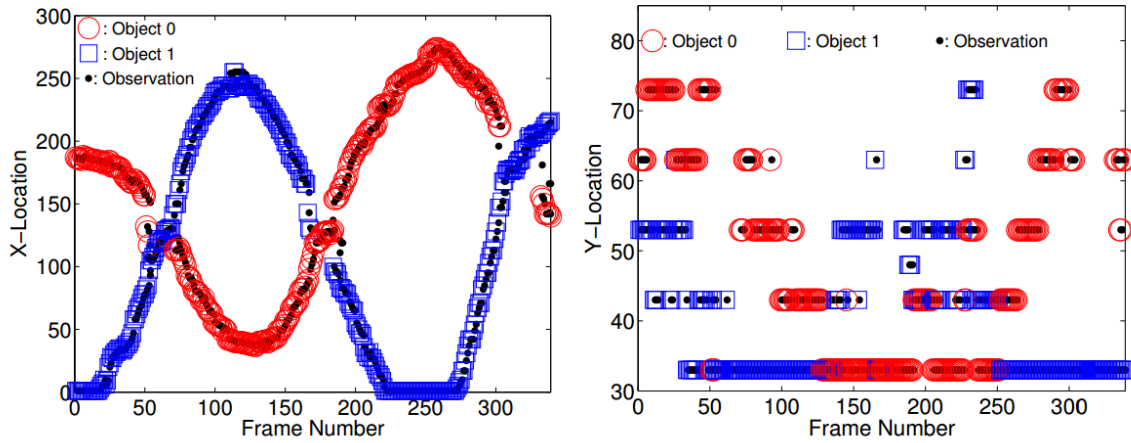


Figure 2.7: Linear Programming method for multi-object tracking [5] The image shows the x and y coordinates of bounding boxes of two moving objects. The global optimum is found using ILP although objects have overlaps in some occasions.

Although LP is optimal, it has some limitations such as modelling occlusions.

- Integer Linear Programming(ILP)

ILP [43] is based on Multiple Shortest Path model in which edges are connected to the paths which is an approach to seek for optimization of all trajectories in all frame sequences. First step is to generate fragments of tracks, namely tracklets, produced by grouping of detected bounding boxes. Then, the tracklets are joint together using Hungarian partitioning algorithm.

In fact it is an approach for finding global optimum and solving the data association problem. First, a network graph is build using the outputs of simple object detectors. Nodes in these graphs are connected to the future and past observations.

ILP tracking formulation used by Pirsiavashi et al. [6] can model occlusion, and is globally optimum.

By representing ILP in a flow network, the shortest path can be found using dynamic programming (figure 2.8).

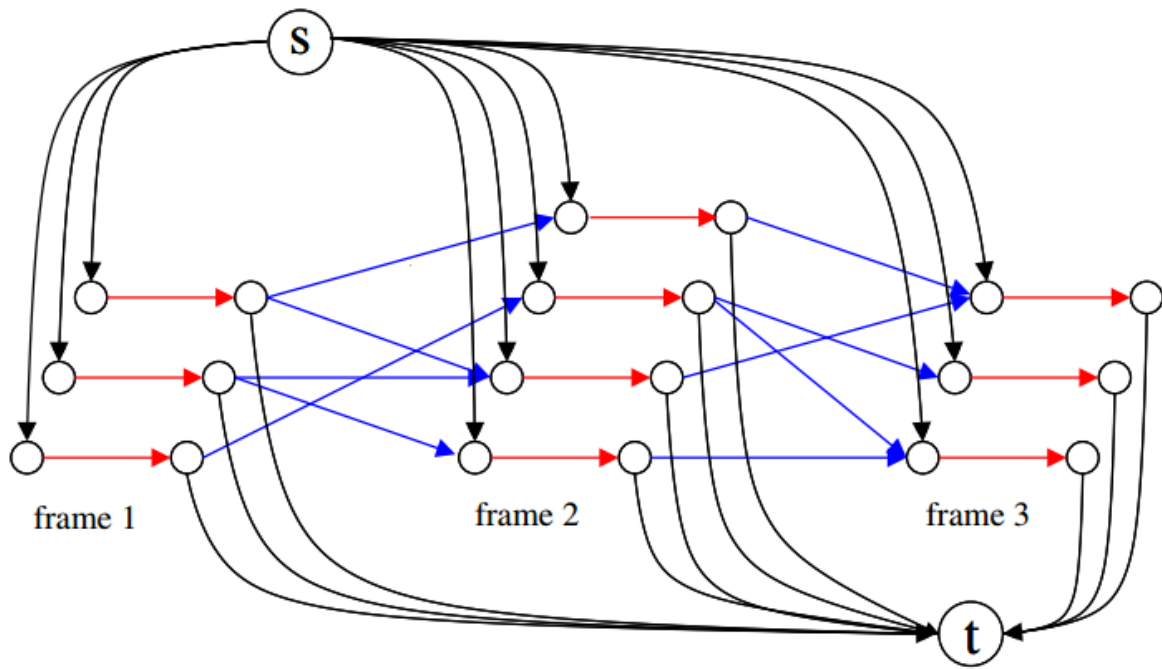


Figure 2.8: Representing ILP in a flow network where the shortest path is used to find the best possible track for each pedestrian.

The occlusion modelling is done by creating tracks from frames which are not successive.

Chapter 3

Methodology

3.1 Detection

Convolutional Neural Networks have performed well on many vision tasks due to the fact that it is inspired by human vision system. Due to advances in General-Purpose Graphics Processing Units (GPGPU) in recent years as well as the availability of large labelled datasets like Caltech[44] and ImageNet[45], training of large CNN have been increasingly feasible.

The main characteristics of CNN are Convolutional and Pooling layers. The former can extract features learned by the network, and the latter allows local transitional invariances in vision tasks. It has been shown that in case of a complicated vision task like image classification and object detection, using features learned by CNN leads to better performance compared to hand-crafted features [2, 46].

To train large CNN using a large dataset is crucial to decrease the chance of overfitting due to the importance of compatibility of the model structure and data shape.

In order to be able to benefit from convolutional neural networks in case of having small dataset, using a pre-trained CNN avoids the problem of overfitting. In the discussed problem in this thesis, only 1000 images containing pedestrians exist to be tracked. Therefore, it was decided to use a Deep convolutional network, DeepPed [10], pre-trained on a large dataset, Caltech

Pedestrian [47], which is a large-scale and widely accepted dataset. Caltech Pedestrian Dataset is about 10 hours of video taken in an urban location by a vehicle driving through the environment. DeepPed is a pedestrian detector algorithm which is based on R-CNN [46]. The main characteristic of this algorithm is combining rich features achieved by a convolutional neural network with region candidates that are likely to be pedestrians. The structure of R-CNN can be seen in the figure 3.1.

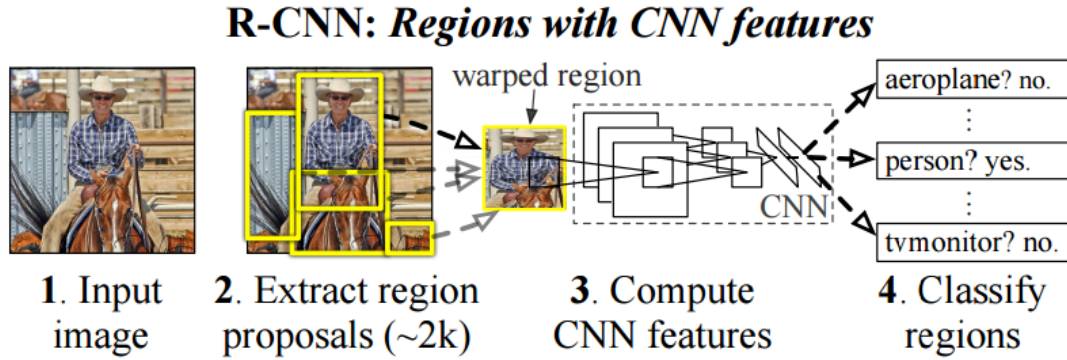


Figure 3.1: R-CNN structure 3.1

In DeepPed, a detection algorithm called Locally Decorralated Channel Features (LDCF) [48], is used which proposes a large set of regions with a confidence value for each region indicating increase the probability of each region to be a pedestrian. The output of DeepPed is a set of bounding box positions as well as their corresponding frame numbers which can be used for tracking.

3.2 Visual Tracking

The process of locating objects in a video is considered as visual tracking. The object may be a moving object like a ball, a person walking in the street, etc.

In many methods of object tracking, objects are initialized manually, or even number of objects in the video is defined. However, in the presented method, there is no need for any initialization, and algorithm can figure it out itself. The tracking step in this work is mostly based on the method presented by Pirsiavashi[6]. In their method, the goal is to (a) find out the number of tracks appear in the sequences and (b) defining the start and end of tracks. Figure 3.2 shows some samples of start and end points in the tracks.

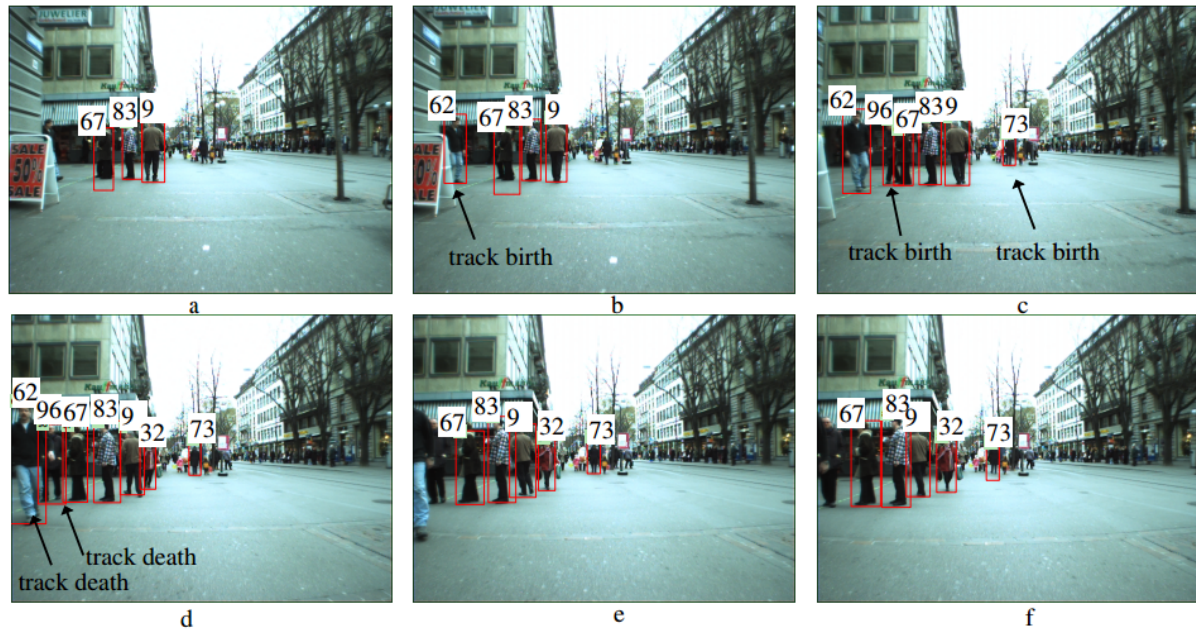


Figure 3.2: This figure shows tracked pedestrians with specified label numbers as well as births and deaths of tracks of individual pedestrians.

In the Pirsiavash work [6], a set of frames in containing bounding boxes detected by a part-based HOG pedestrian detector [41] is assumed as the input. The authors presented a globally optimal multi-object tracking algorithm using a low-level tracker and a graph-based algorithm. Their work is based on the min-cost flow algorithm presented in an analysis of an integer linear programming (ILP) formulation in [43], popular in recent literature. In linear programming, tracking problem is modelled as a multiple-path searching problem and optimizes the detected tracks in terms of finding a consistent track for all the objects, and the locations that objects are occluded. In this work, a cost function is used to solve a multi-object tracking considering the

tracks number, and the birth and death state of the tracks. They presented a method to find the global solution using a greedy algorithm in which tracks are computed by finding the shortest path in a flow network. I have decided to choose this method, since they have used fast and simple algorithms which make it suitable for my goal.

3.3 Post Stage Processing

In order to improve the performance of the tracking results, a post stage processing step after all detection, and tracking steps is proposed. At this level, results consist of both completed tracks as well as broken tracks. The goal in a tracking algorithm is to have smooth tracks containing bounding boxes with the same IDs. Broken tracks are an important issue at this stage leading to a reduction in tracking performance. Broken tracks are short tracks with different IDs containing a few number of bounding boxes. These broken tracks can be joined to the best matches among the other tracks to have smoother tracks and decrease the number of broken tracks. For this purpose, in a set of temporal sequence of frames as the pedestrians move, ends of each track are considered as well as the beginning of all the new tracks started in the next frame. Then, Euclidean distance is used to find the closest detected track starting in the next frame to the ended track in the current frame. In order to make it more precise, a threshold was defined to consider only those bounding boxes which are in the specific threshold area. It is worth mentioning that same threshold values are used in the second experiment on the video recorded at Western..

This approach reduces the number of broken tracks as long as the detection part has a high accuracy in pedestrian detection. However in my case, DeepPed misses some pedestrians in one, two or more than two frames, and this causes broken tracks due to the fact that I am using a different dataset rather than the one that DeepPed was trained with. Simply using Euclidian distance is not enough to distinguish between bounding boxes and find the one which is the continuation of the missed track. For instance, variations in bounding box sizes may confuse

the algorithm to attach tracks to their continuation, and it simply finds the closest bounding box. By only using Euclidian distance for measurement, information about other factors like appearance of targets, motion direction of targets, and size of targets cannot be achieved.

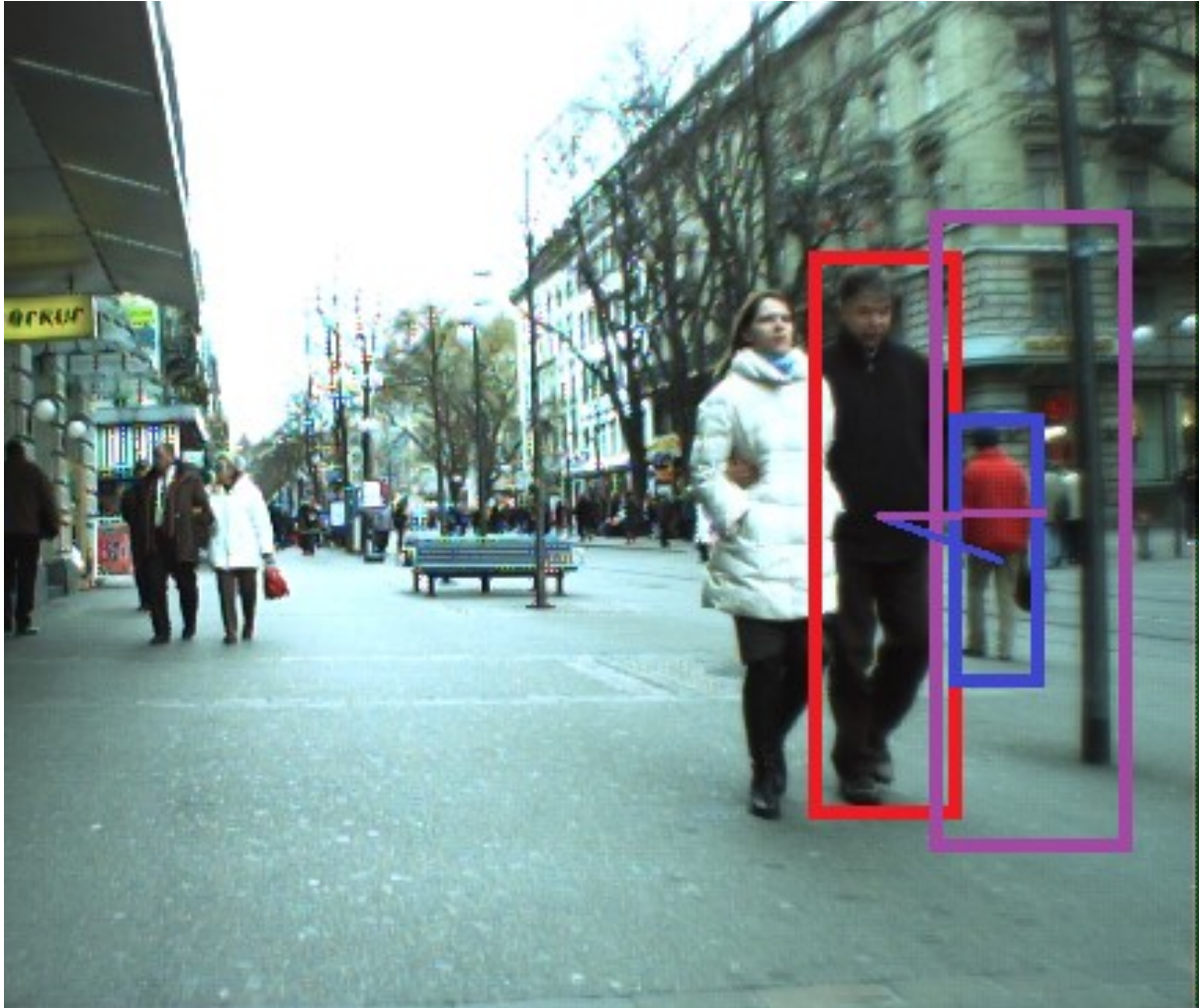


Figure 3.3: Limitation of simply using euclidian distance to join the broken tracks.

Figure 3.3 shows a case where only using Euclidian distance leads to a wrong result. The purple bounding box shows the right location of the person in the red bounding box in the next frame. However, as can be seen in figure 3.3, the blue bounding box is closer to the red bounding box. Therefore, the blue bounding box mistakenly will be considered as the continuation of the desired track. In order to solve this problem, extracting contrast among

the second-order features from each bounding box is proposed to define whether the bounding boxes in each frame are the continuation of the track in the previous frame. In other words, contrast of the current bounding box is computed using equation 3.1 and compared with the contrast value of other candidates.

$$Contrast = \sum_{I_1, I_2} |I_1 - I_2|^2 \log p(I_1, I_2) \quad (3.1)$$

This helps to compare pedestrians in each track and find the best match based on their appearance. Then, a threshold is defined for this part to improve the accuracy. The value of the threshold is defined by trial and error. However, it is worth mentioning that the same values of the thresholds are used in both experiments, on ETHZ and the dataset recorded by myself. It shows that these values are robust enough that they can be used on new datasets containing pedestrians.

However, using only these features is not accurate enough and leads to incorrect detection of the desired bounding box, in cases that a bounding box is found in the desired threshold range of second order features while it is totally far from the target. To solve this issue, a mixture of second order features and Euclidean distance from the centre of each bounding box was used. The flowchart of the proposed Post Stage Processing is shown in figure 3.4. Moreover, finally, the number of false positive tracks are narrowed down by considering a threshold to exclude those tracks containing less than five tracked bounding boxes.

3.4 Algorithm and Software Development Steps

Software environment consisted of MATLAB 2013a running on Ubuntu 14.04 on a desktop which features a Core-i7 3.7 GHz Intel processor, 32 GB RAM , and Geforce GTX 760 GPU. For DeepPed algorithm installation, Caffe, which is a Deep Learning framework, was installed. This Deep Learning library has been developed by the Berkeley Vision and Learning Center

(BVLC). At the first step, Caffe needs a number of prerequisites which should be installed. These include CUDA v7+ (required for GPU mode), BLAS v3.6.0, Boost v1.55, OpenCV v3.0, and IO libraries: lmdb, leveldb. Then Caffe v0.999 package was installed. Since DeepPed is based on R-CNN, the R-CNN package was installed as well. Then, DeepPed package was installed to do the detection phase. After installing DeepPed, MATLAB implementation of the tracking algorithm v1.0 was installed. Then DeepPed was fed with all the frame sequences in the ETHZ dataset in order to detect pedestrians. The output of DeepPed was a set of bounding box positions as well as their corresponding frame numbers which were used for tracking. Then, tracking algorithm was performed on the new data and the tracking results were recorded as a video stream at the end. After all these steps, a Post Stage Processing was implemented to improve the accuracy (Matlab function PostStageFunction is used to perform the proposed Post Stage Processing). It started by finding the end of all the tracks in all of the frames using function TracksSmoother3framesFunc. Then the next frame was checked for the newly started tracks since there was a probability that one of these newly started tracks to be the continuance of the lost track. Therefore, they were compared to the lost track in order to find the best match. Function trackFrame1 finds the best matched track in the next frame. To find the best possible match, first, candidates which were close to the lost tracks were considered using Euclidian distance. Then, among these candidates, the most similar pedestrian to the lost pedestrian in the track was selected based on the similarity of their contrast attribute. In the case of finding the best match, tracks were joined by assigning the same labels. If the best match was not found, two next frame were checked to find the best possible candidate using function trackFrame2. And again, if none of the newly started tracks in the two next frame were matched with the lost track, based on the same Euclidian Distance and contrast conditions, the three next frame were checked for the tracks started in this frame. Function trackFrame3 finds the best possible track in the three next frame. This process is illustrated in the flowchart 3.4. All the code for the post stage processing in MATLAB are provided in Appendix A.

3.5 Dataset

Many datasets have been presented for multi-object tracking using stationary cameras, namely CAVIAR project and PETS2009 datasets [49, 50]. Doing experiments on a dataset with moving cameras is more challenging compared to the stationary ones. In this experiments, ETHZ dataset [51] is used containing both left and right view of a crowded sidewalk which is recorded by cameras attached to a moving stroller. Only the left view of this dataset is used in this work. It consists of four videos each containing about 1000 frames, exported from the video at 14 frames per second. Figure 3.5 shows a sample frame in ETHZ dataset. All the bounding boxes in the dataset are annotated, however, there are no annotations for the track labels. In order to be able to evaluate results of the presented method with the others results, track ID annotations which are defined manually and presented by Milan [52] are used.

I also created a dataset to see the performance of my method on a different input video as well. In order to create this dataset, a camera is fixed in a hallway and recorded about 4 minutes of video, captured from students walking in the hallway. I asked about six students to help with recording this video. It was important to have multiple individuals to be visible in the frames since the suggested algorithm in this thesis works best in tracking multiple pedestrians as shown in figure 3.6. In addition, in this dataset, it is tried to include occlusion as well as distance in this dataset. By distance, it means that this video is recorded in a long hallway to evaluate the proposed method in facing pedestrians which are very small due to their distance from the camera figure 3.7.

3.6 Evaluation

Many methods have been presented in recent years for multiple object tracking, however, there is no specific agreement on the way to evaluate and compare all these methods, and lack of common metrics for evaluation of multiple object tracking algorithms makes it difficult to compare results with the existing algorithms easily. In addition, literature on such evaluation methods is rather sparse. Due to these issues, several methods for multiple object tracking present their results without a quantitative evaluation [53, 54, 55].

Method presented in this work is evaluated by CLEAR-metrics presented by Bernardin et al. [56].

- Multi-Object Tracking Precision (MOTP)

MOTP is the total error which simply computes the average of distance d_t^i between true objects tracked in the ground truth G_t^i and hypothesized tracked objects D_t^i , and c_t , the number of track matches made in frame t . It shows the precision of the tracking algorithm in estimating objects positions without considering the other factors including keeping consistent trajectories (3.2).

$$MOTP = \frac{\sum_{i,t} d_t^i}{\sum_t c_t}, \text{ where } d_t^i = \frac{G_t^i \cap D_t^i}{G_t^i \cup D_t^i} \quad (3.2)$$

- Multi-Object Tracking Accuracy (MOTA)

This method evaluates the multiple object tracking algorithms in terms of accuracy, in which false positives fp_t , missed objects m_t , and ID switches are taken to the account.

$$MOTA = 1 - \frac{\sum (fp_t + m_t + mme_t)}{\sum g_t} \quad (3.3)$$

This formulation is derived from three miss rate error, false positive ratio, and mismatches ratio (3.4), (3.5), and (3.6).

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t} \quad (3.4)$$

$$\overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \quad (3.5)$$

$$\overline{mme} = \frac{\sum_t mme_t}{\sum_t g_t} \quad (3.6)$$

In order to evaluate the presented method, results are compared to the results achieved by the method presented by Pirsiavash [6]. It is tried to consider similar conditions, like using same dataset as well as the same ground truth, for both methods to have a trustable evaluation result.

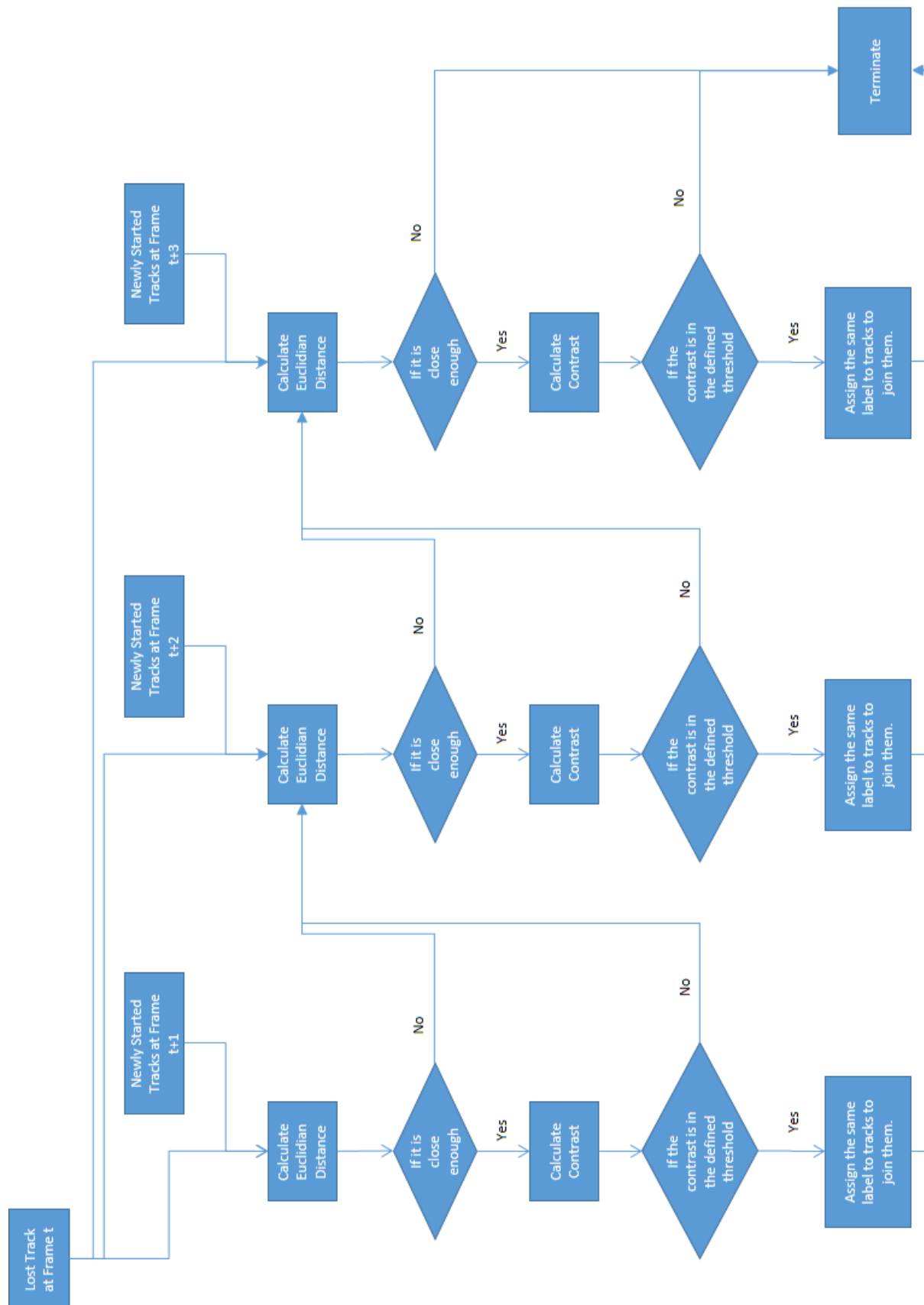


Figure 3.4: Flowchart of the presented Post Stage Processing



Figure 3.5: Sample images from ETHZ Dataset.



Figure 3.6: Sample image from my dataset, showing multiple pedestrians.



Figure 3.7: Sample image from my dataset, showing pedestrians in distance.

Chapter 4

Results

The proposed method is evaluated on the ETHZ as well as a video recorded by myself.

- Detection

As described in section 3.1, the pre-trained deep neural network, DeepPed [10] is used for the detection step. Comparing the suggested detection phase results with the annotations in the original video sequence as well as detection method used in Pirsiavash et al. method [6] shows that DeepPed in many cases is more precise in detecting pedestrians, even the pedestrians that are not annotated in the in the original video sequence. Following figures visually show these differences between results of the proposed algorithm with both the original video sequence and the method used in Pirsiavash et al method [6].

- Tracking

As can be seen in the results in table 4.1, two factors, True Positive Ratio (TPR) as well as False Negative Ratio (FNR) have better results compared to the Pirsiavash et al. method [6]. The Total True Positive Ratio(TPR), False Positive Ratio (FPR), False Negative Ratio (FNR), and True Negative Ratio (TNR) are computed based on the equations 4.1, 4.2, 4.3, and 4.4 respectively.

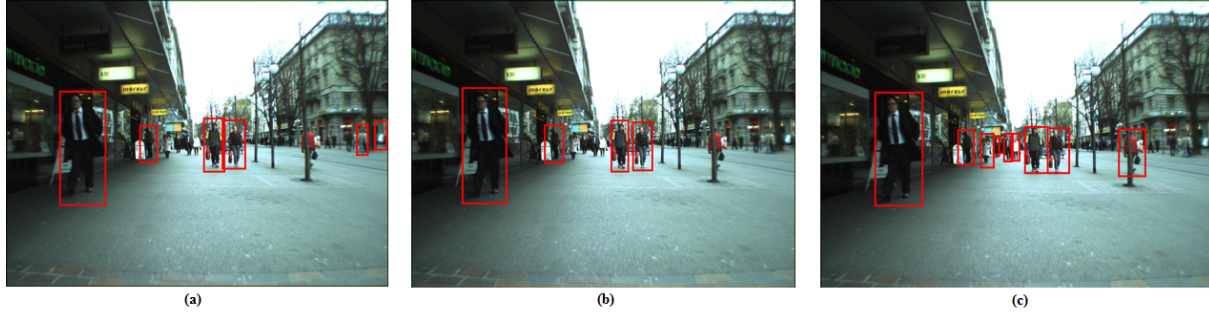


Figure 4.1: (a) represents a specific frame in the tracking results of the proposed method in this thesis. (b) is the same frame which shows the result of tracking method presented by Pirsiavash et al. [6], and (c) shows bounding boxes annotated in the same frame of the original video sequence.

Table 4.1: This table shows the actual True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN) values of the final tracking results in both Pirsiavash et al. method [6], as well as the proposed method.

	TP	FP	FN	TN
Presented Method	4982	1285	2509	6361
Pirsiavash et al. [6]	4870	276	2653	7370

$$TPR = \frac{TP}{(TP + FN)} \quad (4.1)$$

$$FPR = \frac{FP}{(FP + TN)} \quad (4.2)$$

$$FNR = \frac{FN}{(FN + TP)} \quad (4.3)$$

$$TNR = \frac{TN}{(FP + TN)} \quad (4.4)$$

Proposed method is evaluated using Classification of Events, Activities and Relationships (CLEAR) metrics [56], since these metrics are very precise in multi-target tracking evaluation, since they consider important factors such as False Positive, Missed Object, and ID Switch

Table 4.2: This table shows the actual True Positive, False Positive, False Negative, and True Negative Ratio of the final tracking results in both Pirsiavash et al. method [6], as well as the presented method.

	TPR	FPR	FNR	TNR
Proposed Method	66.50%	16.80%	33.49%	83.19%
Pirsiavash et al. [6]	64.73%	3.61%	35.27%	96.39%

Table 4.3: This table shows the actual False Negative, False Positive, as well as ID Switches values in the final tracking results of both Pirsiavash et al. method [6], as well as the proposed method.

	False Negative	False Positive	ID Switch
Proposed Method	2509	1285	155
Pirsiavash et al. [6]	2653	276	123

values. False Positive value shows the rate of those bounding boxes which are not truly pedestrians, while they are considered as pedestrians. Missed Objects rate shows the rate of objects that are truly pedestrian, however they are not detected as pedestrians. The factor of ID Switches is the rate of changing the tracks labels. Two metrics are used in the evaluation of the proposed algorithm: Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). In a multiple tracking problem, for a set of visible objects $\{o_1 \dots o_n\}$ in every frame at time t , a set of hypotheses $\{h_1 \dots h_m\}$ is considered as the tracker output. Table 4.4 reports MOTP, MOTP miss rate, MOTA, and MOTA miss rate of the the presented work compared to the Pirsiavash et al. method [6]. As shown in equation 3.2, MOTP is computed by dividing summation of distances of the proposed tracks using the proposed method, by the total value of True Positives in the annotated dataset. MOTP ratio is calculated using equation 3.3 which considers False Negative, False Positive, and ID Switch values over the total number of pedestrians in the annotated dataset. Table 4.3 shows the the actual values of False Negative, False Positive, as well as ID Switches in both proposed method as well as the Pirsiavash et al. method [6].

As it is shown in table 4.1 and 4.2, improvements are achieved in both TP as well as FN ratios using proposed method. However, due to the increase in the number of False Positives, final results based on MOTA metric is decreased. There are two kinds of false positives in the

Table 4.4: Results of comparing the proposed method with the work introduced by Pirsiavash et al. [6] in terms of MOTP as well as MOTA metrics.

	MOTP	MOTP miss-rate	MOTA	MOTA miss-rate
Proposed Method	70.1%	29.9%	48.35%	51.65%
Pirsiavash et al. [6]	75.25%	24.75%	60.08%	39.92%

results shown in this thesis:

- **Pedestrians which are truly assigned as pedestrians**

In figure 4.3, bounding boxes 231 and 439 show pedestrians which are not annotated in the original video sequences.

Figure 4.4 shows three pedestrians in the right side of the picture which are truly pedestrians, and are detected using the proposed approach, while are not annotated in the original video sequences. In the case that 9 pedestrians are annotated in the original video sequence, finding three false negatives would skew the results, although these pedestrians are truly pedestrians.

Again in figure 4.6, only five pedestrians are annotated in the original video sequences, and having three more detected pedestrians which are considered false positives, would increase the false positive ratio.

In figure 4.7, only three pedestrians are annotated in the original video sequences. However, using the proposed method five pedestrians are detected and tracked.

- **Objects other than pedestrians which are assigned pedestrians**

As can be seen in figure 4.2, although there are some pedestrians in the original video sequences which could not be detected and tracked, bounding box 134 shows a pedestrian detected and tracked which is not annotated in the original video sequences. Besides, bounding box number 413 indicates an accurate false positive. It shows that the proposed method detected and tracked an object instead of a pedestrian.

In figure 4.5, bounding box 51, is detected and tracked. However, bounding box 135 shows an accurate false negative, is considered as a pedestrian while it is the reflection of a pedestrian on a shining surface.

Figure 4.9 also shows another sample of detecting a wrong object instead of pedestrians.

Since annotations in the original video sequence was provided with the dataset was not precise and missed some pedestrians, during testing, any actual pedestrian detected by DeepPed which are not annotated in the original video sequence would be considered as false positives.



Figure 4.2: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 134 and 413 are missclassified although they are truly pedestrians.

There are cases that results exactly match the bounding boxes annotated in the original video sequences. Figure 4.11 shows such a case.

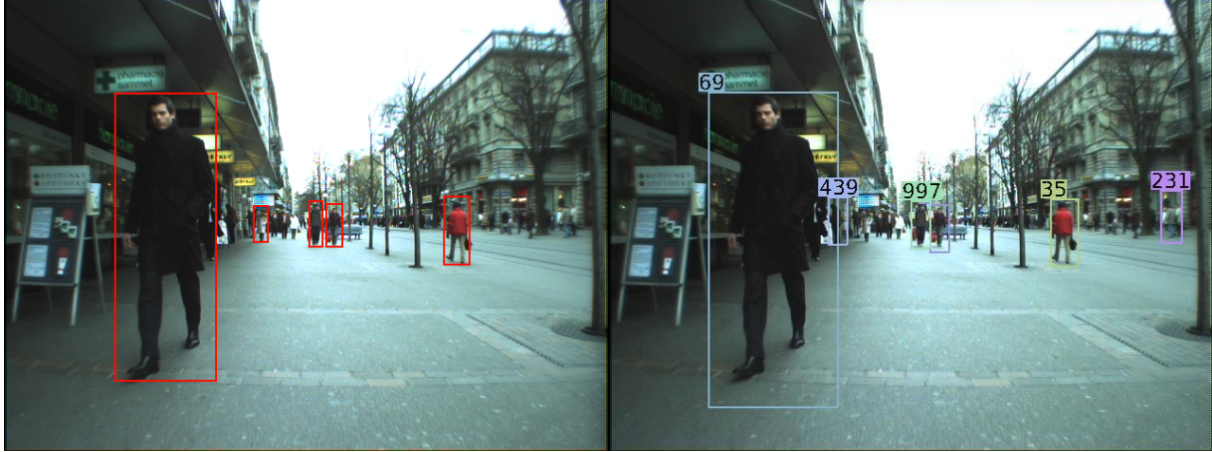


Figure 4.3: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 231 and 439 are missclassified and one pedestrian could not be detected and tracked.

4.1 Occlusion Handling

There are some examples of handling occlusion which can be seen below in figures 4.12 and 4.13 :

4.2 In-house Dataset

As described in section 3.5, automatic object tracking was also performed on a dataset prepared by the author. In figure 4.14, a short period of time in tracking is illustrated. As it can be seen in this figure, multiple pedestrians are tracked, although some of the pedestrians are far away from the camera.

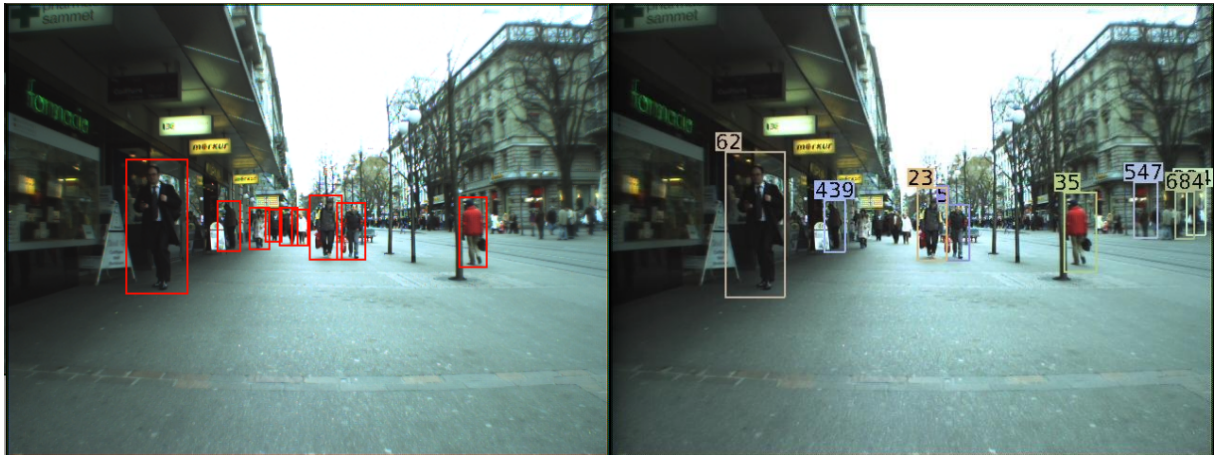


Figure 4.4: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 547 and 684 are missclassified.

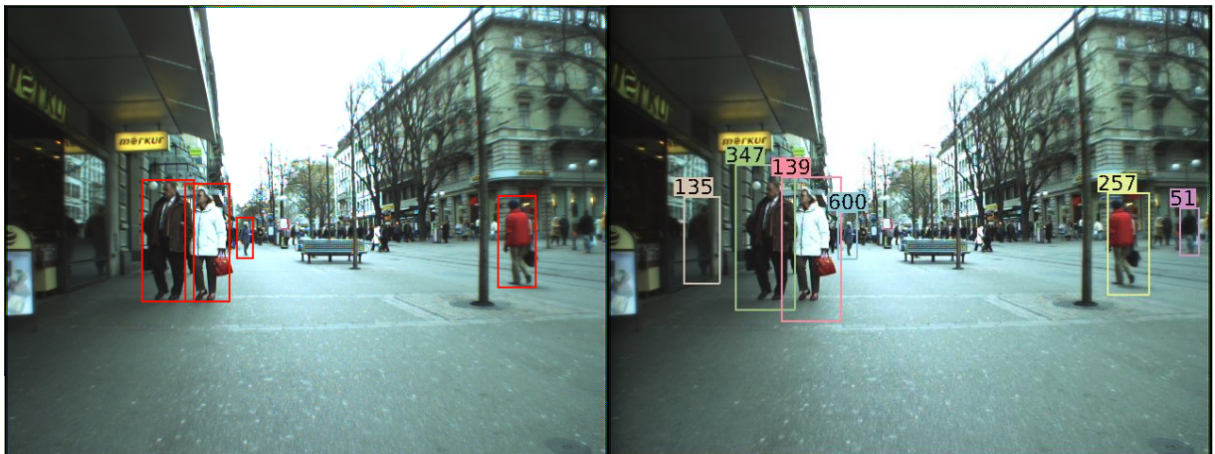


Figure 4.5: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 135 and 51 are missclassified.



Figure 4.6: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. Bounding boxes 244, 312, and another bounding box are missclassified.

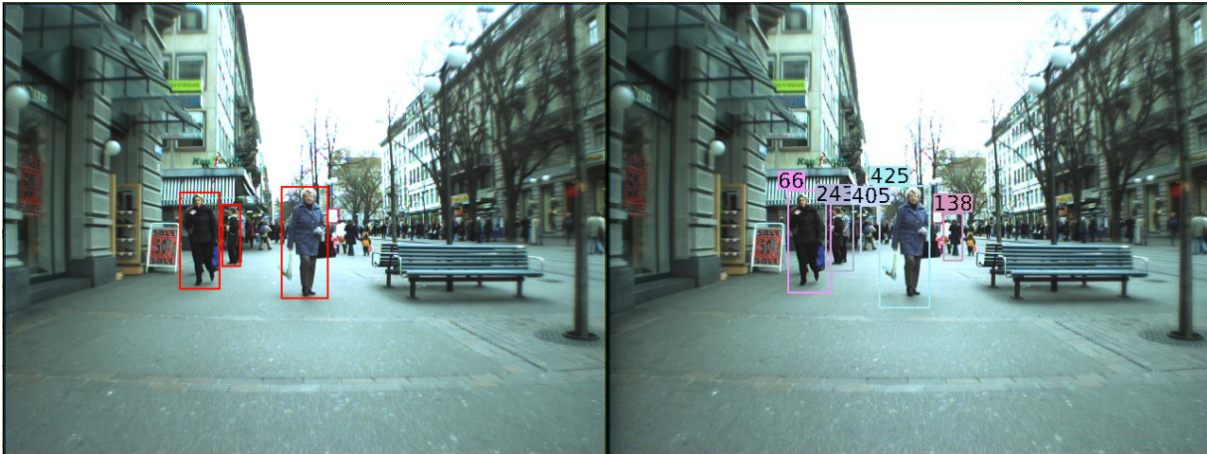


Figure 4.7: Left image shows the annotated bounding boxes in the original video sequences, and the image in right represents the results of proposed method. It shows that the proposed method in some cases is more precise than the annotations in the original video sequence.



Figure 4.8: This is another sample of detecting pedestrians not annotated in the original video sequences.



Figure 4.9: Comparing the original video sequence, on the left, with results achieved by the proposed method, on the right. Bounding boxes 362 and 145 are missclassified although they are truly pedestrians.



Figure 4.10: Comparing the original video sequence, on the left, with results achieved by the proposed method, on the right. Bounding boxes 231 and 684 are missclassified although they are truly pedestrians.

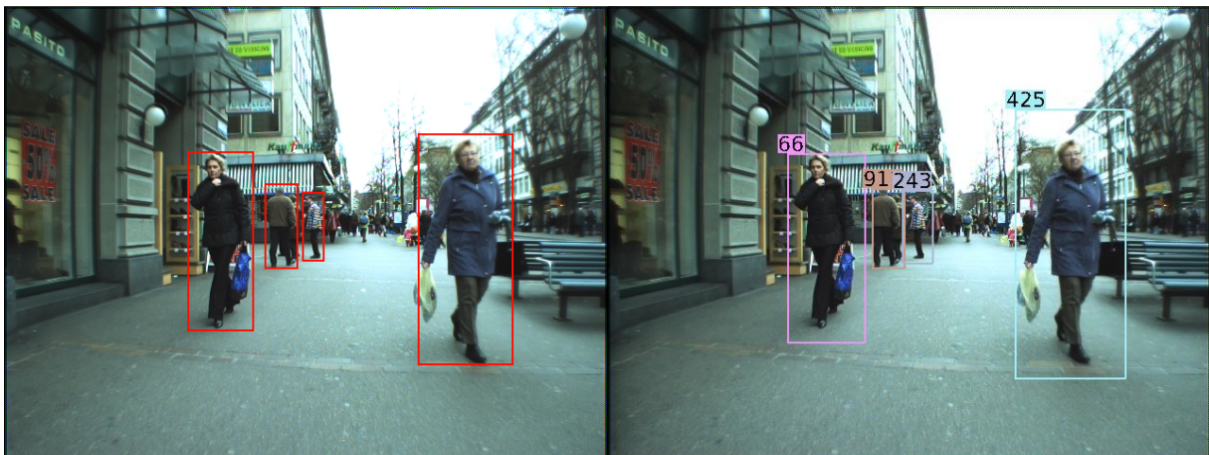


Figure 4.11: This is a sample of equality in the pedestrians detected in both original video sequences and results of the proposed method.

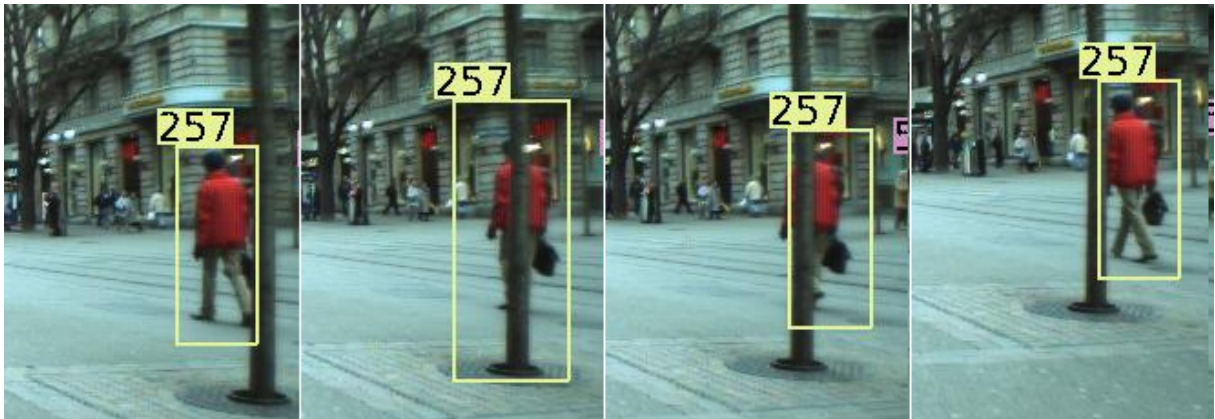


Figure 4.12: A sequence of frames, starting from right to left, in which a pedestrian is occluded with an object. The proposed method handled this problem.



Figure 4.13: Partial Occlusion Handling

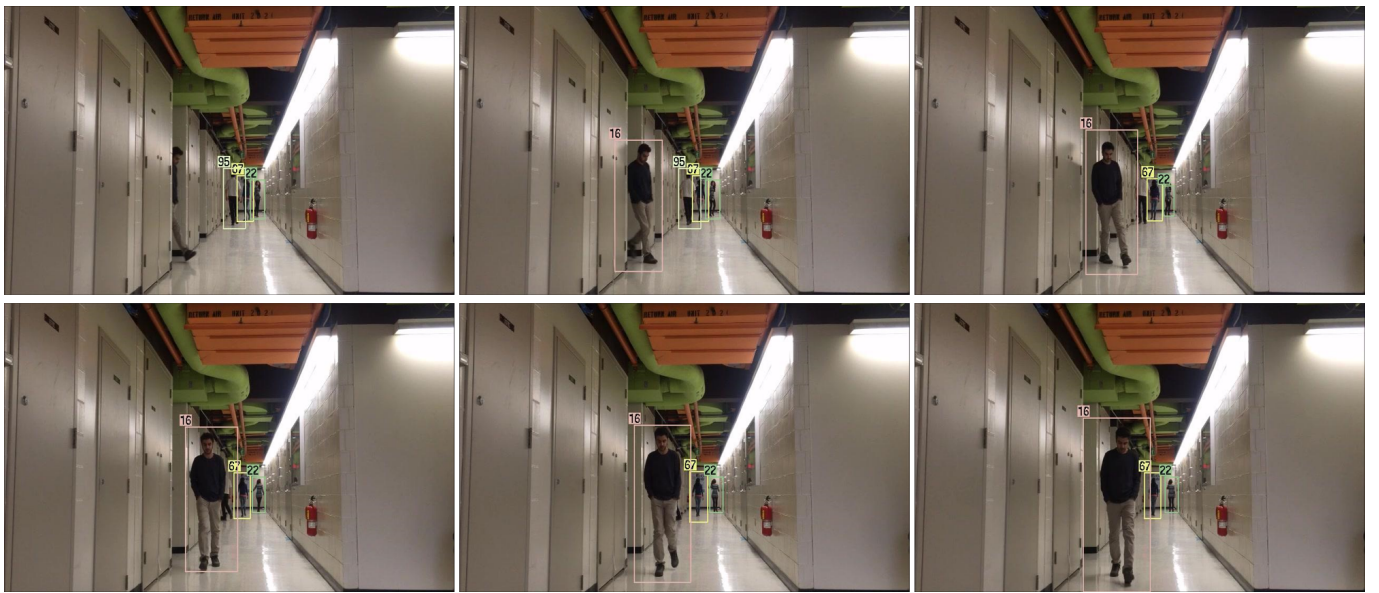


Figure 4.14: These frames show a part of the results of the second experiment presented in this study. The same approach as the first experiment on ETHZ dataset is used using same values for all the parameters like Euclidian distance threshold.

Chapter 5

Discussion and Future Work

5.1 Discussion

In the proposed work, an automated multiple-pedestrian tracking method is presented. This method benefits from a state-of-the-art deep CNN and a tracking algorithm to track people walking in the street. A novel post stage is also presented in order to improve the performance of the proposed tracking method. For the 999 frames in ETHZ dataset, in MATLAB R2014b, the whole process of detection takes 1935.441385 seconds to run, and the tracking step takes 36.178885 seconds on a desktop which features a Core-i7 3.7 GHz Intel processor, 32 GB RAM, and Geforce GTX 760 GPU.

The detection algorithm was performed with GPU acceleration using the OpenCV library [57] to increase the processing speed. Pre-training of DeepPed was performed by Tom et al. [10] in Caffe which is an open source library [2]. This research presents a combination of a detection and tracking methods for the problem of multi-pedestrian tracking. A novel post stage processing algorithm was used to increase the accuracy in tracking pedestrians. Using Deep CNN in the detection task and causing more accurate detection, paves the way to achieve more accurate results in tracking. The method presented in this work is compared with a tracking method presented by Pirsiavash et al. [6]

Results showed improvement in both finding true pedestrians (true positives) as well as decreasing the rate of false negatives. However, there was an increase in false positives ratio which shows that the algorithm sometimes tracked objects other than pedestrians. Upon closer examination of the video sequence, it was observed that this was mostly due to the fact that the proposed method finds many true pedestrians which are not annotated in the original video sequence. Therefore, the accuracy analysis calculation considered these types of differences between the achieved results and the original video sequence as false positives, leading to an artificially increased false positive ratio.

5.2 Future work

Proposed method can be evaluated more accurately if such instances of unlabelled pedestrians are added to the ground truth provided with the original dataset. This will require annotating all the frames in the ETHZ dataset with both bounding boxes and labels, in the way that it contain all the unlabelled pedestrians. However this will be a very time consuming process and it was decided to postpone this task.

There are several ways to improve the proposed post stage processing algorithm. Firstly, instead of considering the next three sequences of frames to check the occlusion problem, longer sequences may be taken into the account. The other way to improve the accuracy is to find the best values for the thresholds in both distance and second order features. However, such an approach is complicated by the fact that different frames will require different values of these tuning parameters because of the changing characteristics. For example, in terms of distance, the threshold would be different in the frame number 1 compared to frame number 3. In this example, pedestrians who are visible immediately in the next frame may be closer to each other compared to the pedestrians visible in the next two frame.

Finally, HOG Person Detector may be used instead of second order features that was used to find the most similar pedestrian to the current one to explore whether that would increase the

accuracy of track merging.

Bibliography

- [1] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM computing surveys (CSUR)*, 38(4):13, 2006.
- [2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [3] Joseph Rickert. Revolutions, h2o world 2015. <http://blog.revolutionanalytics.com/2015/11/h2o-world-2015.html>, 2015. [Online; accessed 15-Dec-2015].
- [4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
- [5] Hao Jiang, Sidney Fels, and James J Little. A linear programming approach for multiple object tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [6] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208. IEEE, 2011.
- [7] Photo Review magazine. How useful is in-camera face detection? <http://www.photoreview.com.au/tips/shooting/>

- insider-how-useful-is-in-camera-face-detection, 2016. [Online; accessed 3-July-2016].
- [8] Jaideep Jeyakar, R Venkatesh Babu, and KR Ramakrishnan. Robust object tracking using local kernels and background information. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V–49. IEEE, 2007.
- [9] Qi Zhao, Jinman Kang, Hai Tao, and Wei Hua. Part based human tracking in a multiple cues fusion framework. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 1, pages 450–455. IEEE, 2006.
- [10] Denis Tom, Federico Monti, Luca Baroffio, Luca Bondi, Marco Tagliasacchi, and Stefano Tubaro. Deep convolutional neural networks for pedestrian detection. *arXiv preprint arXiv:1510.03608*, 2015.
- [11] Lu Jin, Shenghua Gao, Zechao Li, and Jinhui Tang. Hand-crafted features or machine learnt features? together they improve rgb-d object recognition. In *Multimedia (ISM), 2014 IEEE International Symposium on*, pages 311–319. IEEE, 2014.
- [12] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.
- [13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.
- [14] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2903–2910. IEEE, 2012.

- [15] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1532–1545, 2014.
- [16] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014.
- [17] Rainer Lienhart, Luhong Liang, and Alexander Kuranov. A detector tree of boosted classifiers for real-time object detection and tracking. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 2, pages II–277. IEEE, 2003.
- [18] Piotr Dollár, Ron Appel, and Wolf Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Computer Vision–ECCV 2012*, pages 645–659. Springer, 2012.
- [19] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [20] Andrzej Materka, Michal Strzelecki, et al. Texture analysis methods—a review. *Technical university of lodz, institute of electronics, COST B11 report, Brussels*, pages 9–11, 1998.
- [21] A Ardeshir Goshtasby. *Image registration: Principles, tools and methods*. Springer Science & Business Media, 2012.
- [22] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.
- [23] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

- [24] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [25] Chang-Yuan Cheng, Kuang-Hui Lin, and Chih-Wen Shih. Multistability in recurrent neural networks. *SIAM Journal on Applied Mathematics*, 66(4):1301–1320, 2006.
- [26] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.
- [27] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.
- [28] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.
- [29] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.
- [30] N Senthilkumaran and R Rajesh. Edge detection techniques for image segmentation—a survey of soft computing approaches. *International Journal of Recent Trends in Engineering and Technology*, 1(2), 2009.
- [31] Massimo Piccardi. Background subtraction techniques: a review. In *Systems, man and cybernetics, 2004 IEEE international conference on*, volume 4, pages 3099–3104. IEEE, 2004.
- [32] Wanli Ouyang and Xiaogang Wang. Joint deep learning for pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2056–2063, 2013.

- [33] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2013.
- [34] Wanli Ouyang and Xiaogang Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3258–3265. IEEE, 2012.
- [35] Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Pedestrian detection aided by deep learning semantic tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5079–5087, 2015.
- [36] Zia Khan, Tucker Balch, and Frank Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(11):1805–1819, 2005.
- [37] Bo Wu and Ram Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 951–958. IEEE, 2006.
- [38] Jérôme Bercla, Francois Fleuret, and Pascal Fua. Robust people tracking with global trajectory optimization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 744–750. IEEE, 2006.
- [39] Bastian Leibe, Konrad Schindler, and Luc Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [40] Qian Yu, Gérard Medioni, and Isaac Cohen. Multiple target tracking using spatio-temporal markov chain monte carlo data association. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

- [41] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [42] Matthew Rowan and Frederic D Maire. An efficient multiple object vision tracking system using bipartite graph matching. FIRA Robot World Congress, 2004.
- [43] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
- [44] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):594–611, 2006.
- [45] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [46] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [47] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.
- [48] Woonhyun Nam, Piotr Dollar, and Joon Hee Han. Local decorrelation for improved pedestrian detection. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 424–432. Curran Associates, Inc., 2014.

- [49] Robert B Fisher. The pets04 surveillance ground-truth data sets. In *Proc. 6th IEEE international workshop on performance evaluation of tracking and surveillance*, pages 1–5, 2004.
- [50] JAMES Ferryman, J Crowley, and A Shahrokni. Pets 2009 benchmark data. In *Computer Vision and Pattern Recognition, Miami, Florida: IEEE Conference on*, 2009.
- [51] Andreas Ess, Bastian Leibe, and Luc Van Gool. Depth and appearance for mobile scene analysis. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [52] Anton Milan. Ground truth annotations. <http://milanton.de/data.html>. [Online; accessed 2-Feb-2016].
- [53] Wei Niu, Long Jiao, Dan Han, and Yuan-Fang Wang. Real-time multiperson tracking in video surveillance. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, volume 2, pages 1144–1148. IEEE, 2003.
- [54] Ismail Haritaoglu, David Harwood, and Larry S Davis. W4s: A real-time system for detecting and tracking people in 2 1/2d. In *Computer VisionECCV'98*, pages 877–892. Springer, 1998.
- [55] Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfindex: Real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):780–785, 1997.
- [56] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *Journal on Image and Video Processing*, 2008:1, 2008.
- [57] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

Appendix A

MATLAB Implementation

% This function is used in Post Stage Processing in order to solve the broken
→ tracks problem.

```
function bboxes_tracked = PostStage(bboxes_tracked)

j=0;

for i = 1 :size(bboxes_tracked,2)
    if ~(isempty(bboxes_tracked(i).bbox(:,:)))
        j=j+1;
        tempt_bboxes_tracked(j).bbox = bboxes_tracked(i).bbox ;
    end
end

bboxes_tracked=[];

bboxes_tracked=tempt_bboxes_tracked;

for i = 1 : size(bboxes_tracked,2)
    c=size(bboxes_tracked(i).bbox,1);
    for x= 1:c
        bboxes_tracked(i).bbox(:,6)=0; % tracked flag
        bboxes_tracked(i).bbox(:,7)=0; % start of track flag
    end
end
```



```

        if isempty(CurrentBboxInd)
            bboxes_tracked(i).bbox(x,8)=1;
        else
            bboxes_tracked(j-1).bbox(CurrentBboxInd,8)=1; % set the
                ↪ flag to 1 for end of track
        end

    end

    CurrentBboxInd=k;
    ind(count) = bboxes_tracked(i).bbox(x,5);
    j=j+1;
    count=count+1;
end

end

end

bboxes_tracked = TracksSmoother3framesFunc(bboxes_tracked);

for i = 1 : size(bboxes_tracked,2)
    bboxes_tracked(i).bbox= bboxes_tracked(i).bbox(:,1:5);
end

```

% This function contributes to solve the broken tracks to have smoother
 ↪ tracks considering three next frames after each lost track.

```

function bboxes_tracked = TracksSmoother3framesFunc(bboxes_tracked)

for i=1:size(bboxes_tracked,2)-3

```

```

if isempty(bboxes_tracked(i).bbox)
    continue;
end
[endInd, endVal] = find(bboxes_tracked(i).bbox(:,8)==1);
if isempty(endInd)
    continue;
end
for c = 1 : size (endInd,1)
    if bboxes_tracked(i).bbox(endInd(c),9)==0
        bboxes_tracked(i).bbox(endInd(c),9)=1;
        TrackbackId = bboxes_tracked(i).bbox(endInd(c),5);
        preFrame=i;
        k=1;
        while ~(isempty(k)) && (preFrame>2)
            preFrame=preFrame-1;
            k = find(bboxes_tracked(preFrame).bbox(:,5) == TrackbackId);
            bboxes_tracked(preFrame).bbox(k,9)=1;
        end
        input_frames = 'data/testImage/image_\'%0.8d_0.png';
%         input_frames = 'data/testImage/MyDsMovie %0.4d.jpg';
        img= imread(sprintf(input_frames, i));
        im= img(round(bboxes_tracked(i).bbox(endInd(c),2)):
            ↪ round(bboxes_tracked(i).bbox(endInd(c),4)),
            ↪ round(bboxes_tracked(i).bbox(endInd(c),1)):
            ↪ round(bboxes_tracked(i).bbox(endInd(c),3)),:));

        x = (bboxes_tracked(i).bbox(endInd(c),1)+
            ↪ bboxes_tracked(i).bbox(endInd(c),3))/2;

```

```

y = (bboxes_tracked(i).bbox(endInd(c),2)+
    ↪ bboxes_tracked(i).bbox(endInd(c),4))/2;
TrackID = bboxes_tracked(i).bbox(endInd(c),5); % Track Id

[startInd1nx, startVal] = find(bboxes_tracked(i+1).bbox(:,7)== 1);
[startInd2nx, startVal] = find(bboxes_tracked(i+2).bbox(:,7)== 1);
[startInd3nx, startVal] = find(bboxes_tracked(i+3).bbox(:,7)== 1);

flg=0;

if isempty(startInd1nx)
    if isempty(startInd2nx)
        if isempty(startInd3nx)
            continue;
        else
            [bboxes_tracked, flg] = trackFrame3(im, x, y,
                ↪ bboxes_tracked, i, startInd3nx, TrackID, endInd,
                ↪ c);
        end
    else
        [bboxes_tracked, flg] = trackFrame2(im, x, y,
            ↪ bboxes_tracked, i, startInd2nx, TrackID, endInd, c);
        if flg==0
            if isempty(startInd3nx)
                continue;
            else
                [bboxes_tracked, flg] = trackFrame3(im, x, y,
                    ↪ bboxes_tracked, i, startInd3nx, TrackID,
                    ↪ endInd,c);
            end
        end
    end
end

```

```

        end
    end
end
else
    [bboxes_tracked, flg] = trackFrame1(im, x, y, bboxes_tracked,
        ↪ i, startInd1nx, TrackID, endInd, c);
    if flg==0
        if isempty(startInd2nx)
            if isempty(startInd3nx)
                continue;
            else
                [bboxes_tracked, flg] = trackFrame3(im, x, y,
                    ↪ bboxes_tracked, i, startInd3nx, TrackID,
                    ↪ endInd, c);
            end
        else
            [bboxes_tracked, flg] = trackFrame2(im, x, y,
                ↪ bboxes_tracked, i, startInd2nx, TrackID, endInd,
                ↪ c);
            if flg==0
                if isempty(startInd3nx)
                    continue;
                else
                    [bboxes_tracked, flg] = trackFrame3(im, x, y,
                        ↪ bboxes_tracked, i, startInd3nx, TrackID,
                        ↪ endInd, c);
                end
            end
        end
    end
end
end

```

```

        end
    end
end
end
end
end

% This function looks at the next frame to see if there are any new tracks
% started at that frame, if so, then it uses feature extraction( second
    → order) to define whether the bounding box in
% this frame is the continuation of the track in the previous frame.

function [bboxes_tracked,flg]=trackFrame1(im_i, x, y, bboxes_tracked, i,
    → startInd1nx, TrackID, endInd, c)

xx=(bboxes_tracked(i+1).bbox(startInd1nx,1)+
    → bboxes_tracked(i+1).bbox(startInd1nx,3))/2;
yy=(bboxes_tracked(i+1).bbox(startInd1nx,2)+
    → bboxes_tracked(i+1).bbox(startInd1nx,4))/2;
distancePoints = sqrt((xx-x).^2+(yy-y).^2);
if tempDistance1nx>36
    flg=0;
    return;
else
%   input_frames = 'data/testImage/MyDsMovie %0.4d.jpg';
    input_frames = 'data/testImage/image_\'%0.8d_0.png';
    im_i_featureVector=featureExtraction(im_i);
    img= imread(sprintf(input_frames, i+1));
    x_i_size= size(im_i,1);

```

```

y_i_size= size(im_i,2);
im= img(round(bboxes_tracked(i+1).bbox(startInd1nx(indsDistance1nx),2)):
    ↪ round(bboxes_tracked(i+1).bbox(startInd1nx(indsDistance1nx),4)),
    ↪ round(bboxes_tracked(i+1).bbox(startInd1nx(indsDistance1nx),1)):
    ↪ round(bboxes_tracked(i+1).bbox(startInd1nx(indsDistance1nx),3)),:));
im=imresize(im, [x_i_size y_i_size]);
im_featureVector=featureExtraction(im);
E_distance=histogramComparision(im_i,im);
distPoints = sqrt(sum((im_i_featureVector - im_featureVector) .^ 2));
if distPoints>0.76 %0.76 for second order, 14 for first order
    %    if E_distance>0.07
    flg=0;
    return;
end
ClosestTrackID = bboxes_tracked(i+1).bbox(startInd1nx(indsDistance1nx),5);
    ↪ % the closest frame's Track Id
k = 1;
nextFrame=i;
flg=1;

while ~(isempty(k)) && (nextFrame<size(bboxes_tracked,2))
    nextFrame=nextFrame+1;
    k = find(bboxes_tracked(nextFrame).bbox(:,5) == ClosestTrackID);
    if bboxes_tracked(nextFrame).bbox(k,9)==0
        bboxes_tracked(nextFrame).bbox(k,5)= TrackID;
        bboxes_tracked(nextFrame).bbox(k,9)=1;
    end
end
end
end

```

end

```
% This function looks at the two-next frame to see if there are any new tracks
% started at that frame, if so, then it uses feature extraction( second
    ↪ order) to define whether the bounding box in
% this frame is the continuation of the track in the two-previous frame.
```

```
function [bboxes_tracked,flg]=trackFrame2(im_i, x, y, bboxes_tracked, i,
    ↪ startInd2nx, TrackID, endInd, c)
```

```
xx=(bboxes_tracked(i+2).bbox(startInd2nx,1)+
    ↪ bboxes_tracked(i+2).bbox(startInd2nx,3))/2;
yy=(bboxes_tracked(i+2).bbox(startInd2nx,2)+
    ↪ bboxes_tracked(i+2).bbox(startInd2nx,4))/2;
distancePoints = sqrt((xx-x).^2+(yy-y).^2);
[tempDistance2nx, indsDistance2nx]=min(distancePoints);
if tempDistance2nx>45
    flg=0;
    return;
```

```
else
```

```
    input_frames = 'data/testImage/image\_%0.8d_0.png';
%    input_frames = 'data/testImage/MyDsMovie %0.4d.jpg';
    im_i_featureVector=featureExtraction(im_i);
    img= imread(sprintf(input_frames, i+2));
    x_i_size= size(im_i,1);
    y_i_size= size(im_i,2);
    im= img(round(bboxes_tracked(i+2).bbox(startInd2nx(indsDistance2nx),2)):
        ↪ round(bboxes_tracked(i+2).bbox(startInd2nx(indsDistance2nx),4)),
        ↪ round(bboxes_tracked(i+2).bbox(startInd2nx(indsDistance2nx),1))):
```



```

    ↪ round(bboxes_tracked(i+2).bbox(startInd2nx(indsDistance2nx),3)),:);
im=imresize(im, [x_i_size y_i_size]);
im_featureVector=featureExtraction(im);
distPoints = sqrt(sum((im_i_featureVector - im_featureVector) .^ 2));
E_distance=histogramComparision(im_i,im);
if E_distance>0.07
    flg=0;
    return;
end

ClosestTrackID = bboxes_tracked(i+2).bbox(
    ↪ startInd2nx(indsDistance2nx),5); % the closest frame's Track Id
bboxes_tracked(i+1).bbox= [bboxes_tracked(i+1).bbox;
    ↪ bboxes_tracked(i).bbox(endInd(c),:)]];
bboxes_tracked(i+1).bbox(end,7)=0;
bboxes_tracked(i+1).bbox(end,8)=0;

k = 1;
nextFrame=i+1;
flg=1;

while ~(isempty(k)) && (nextFrame<size(bboxes_tracked,2))
    nextFrame=nextFrame+1;
    k = find(bboxes_tracked(nextFrame).bbox(:,5) == ClosestTrackID);
    if bboxes_tracked(nextFrame).bbox(k,9)==0
        bboxes_tracked(nextFrame).bbox(k,5)= TrackID;
        bboxes_tracked(nextFrame).bbox(k,9)=1;
    end
end
end

```

end

end

% This function looks at the three-next frame to see if there are any new

→ tracks

% started at that frame, if so, then it uses feature extraction(second

→ order) to define whether the bounding box in

% this frame is the continuation of the track in the three-previous frame.

function [bboxes_tracked,flg]=trackFrame3(im_i, x, y, bboxes_tracked, i,

→ startInd3nx, TrackID, endInd, c)

input_frames = 'data/testImage/image_\'%0.8d_0.png';

% input_frames = 'data/testImage/MyDsMovie %0.4d.jpg';

im_i_featureVector=featureExtraction(im_i);

img= imread(sprintf(input_frames, i+3));

x_i_size= size(im_i,1);

y_i_size= size(im_i,2);

xx=(bboxes_tracked(i+3).bbox(startInd3nx,1)+

→ bboxes_tracked(i+3).bbox(startInd3nx,3))/2;

yy=(bboxes_tracked(i+3).bbox(startInd3nx,2)+

→ bboxes_tracked(i+3).bbox(startInd3nx,4))/2;

distancePoints = sqrt((xx-x).^2+(yy-y).^2);

[tempDistance3nx, indsDistance3nx]=min(distancePoints);

if tempDistance3nx>60

flg=0;

return;

end

im= img(round(bboxes_tracked(i+3).bbox(startInd3nx(indsDistance3nx),2)):

```

    ↪ round(bboxes_tracked(i+3).bbox(startInd3nx(indsDistance3nx),4)),
    ↪ round(bboxes_tracked(i+3).bbox(startInd3nx(indsDistance3nx),1)):
    ↪ round(bboxes_tracked(i+3).bbox(startInd3nx(indsDistance3nx),3)),:);
im=imresize(im, [x_i_size y_i_size]);
im_featureVector=featureExtraction(im);
distPoints = sqrt(sum((im_i_featureVector - im_featureVector) .^ 2));
E_distance=histogramComparision(im_i,im);
if E_distance>0.07
    flg=0;
    return;
end
ClosestTrackID = bboxes_tracked(i+3).bbox( startInd3nx(indsDistance3nx),5); %
    ↪ the closest frame's Track Id
bboxes_tracked(i+1).bbox= [bboxes_tracked(i+1).bbox;
    ↪ bboxes_tracked(i).bbox(endInd(c),:)]];
bboxes_tracked(i+1).bbox(end,7)=0;
bboxes_tracked(i+1).bbox(end,8)=0;
bboxes_tracked(i+2).bbox= [bboxes_tracked(i+2).bbox;
    ↪ bboxes_tracked(i).bbox(endInd(c),:)]];
bboxes_tracked(i+2).bbox(end,7)=0;
bboxes_tracked(i+2).bbox(end,8)=0;
flg=1;
k = 1;
nextFrame=i+2;
while ~(isempty(k)) && (nextFrame<size(bboxes_tracked,2))
    nextFrame=nextFrame+1;
    k = find(bboxes_tracked(nextFrame).bbox(:,5) == ClosestTrackID);
    if bboxes_tracked(nextFrame).bbox(k,9)==0
        bboxes_tracked(nextFrame).bbox(k,5)= TrackID;

```

```

        bboxes_tracked(nextFrame).bbox(k,9)=1;
    end
end
end

```

```

% This function omits those tracks which are less probable to be truly tracks.
function bboxes_tracked=omitSpareBBoxes(bboxes_tracked)
bboxes_tracked_=bboxes_tracked;
for i = 1 : size(bboxes_tracked_,2)
    c=size(bboxes_tracked_(i).bbox,1);
    for x= 1:c
        bboxes_tracked_(i).bbox(:,6)=0; % tracked flag
    end
end
for i = 1 : size(bboxes_tracked_,2)
    for j = 1 : size(bboxes_tracked_(i).bbox,1)
        id = bboxes_tracked_(i).bbox(j,5);
        if bboxes_tracked_(i).bbox(j,6)==0 % if it has not been tracked yet
            bboxes_tracked_(i).bbox(j,6)=1;
            k = 1;
            nextFrame=i;
            count=0;
            while ~(isempty(k)) && (nextFrame<size(bboxes_tracked,2))
                nextFrame=nextFrame+1;
                n=k;
                k = find(bboxes_tracked_(nextFrame).bbox(:,5) == id);
                bboxes_tracked_(nextFrame).bbox(k,6)=1;
                count=count+1;
            end

```

```
    if count<5
        while ~(isempty(n)) && (nextFrame>1)
            nextFrame=nextFrame-1;
            n = find(bboxes_tracked_(nextFrame).bbox(:,5) == id);
            bboxes_tracked_(nextFrame).bbox(n,:)=0;
        end
    end
    if count==1
        bboxes_tracked_(i).bbox(j,:)=0;
    end
end
end
end

for i = 1: size(bboxes_tracked,2)
    n = find(bboxes_tracked_(i).bbox(:,6)==0);
    bboxes_tracked_(i).bbox(n,:)=[];
    bboxes_tracked(i).bbox= bboxes_tracked_(i).bbox(:,1:5);
end
end
```

Curriculum Vitae

Name: Marjan Ramin

Post-Secondary Education and Degrees: Shahid Rajaee University
Tehran, Iran
BSc in Software Engineering

University of Western Ontario
London, ON
MESc in Software Engineering

Experience: The University of Western Ontario
2014 - 2016